21世纪

高等院校规划教材

邓安文 编著

密码学——加密演算法



中国水利水电出版社
www.waterpub.com.cn

责任编辑：朱江浩　　封面设计：俞兆君

- 采用“任务驱动”的编写方式，引入案例和启发式教学方法
- 提供电子教案、案例素材等教学资源，教材立体化配套
- 满足高等院校应用型人才培养的需要

大学计算机基础
办公自动化实用技术
计算机科学导论
计算机专业英语
C语言程序设计
C++程序设计
C#程序设计
SQL Server 2000数据库及应用
Visual C++程序设计及应用
Visual Basic 6.0程序设计及应用
Visual Basic.NET程序设计
Visual FoxPro 6.0程序设计及应用
Java程序设计及应用
ASP程序设计及应用
ASP.NET程序设计及应用
PowerBuilder程序设计及应用
Web程序设计及应用
数据结构（C++语言描述）
数据结构（C语言描述）
微机原理与汇编语言
微型计算机原理与接口技术
汇编语言程序设计
操作系统原理及应用
Access 2002数据库及应用
大学信息技术概论
数据库原理及应用（SQL Server 2000）
数据库原理及开发（SQL Server+PowerBuilder）
Web数据库与XML应用
网页设计与制作实用技术
动态网页设计实用技术
移动通信
电子技术
电工技术
网站建设与管理
数据通信与计算机网络
密码学—加密演算法
计算机网络实用技术
局域网技术与组网工程
计算机网络安全技术
综合布线技术与施工
网络互联技术
单片机原理及应用
DSP原理及应用
软件工程
微机组装与维护实用技术
多媒体技术与应用
计算机控制与仿真技术
电子商务概论
管理信息系统
自动控制原理

北京万水电子信息有限公司
Beijing Multi-Channel Electronic Information Co.,Ltd.
地址：北京市海淀区长春桥路5号新起点嘉园4号楼1706室
电话：010-82562819(总机)
传真：010-82564371
邮编：100089
E-mail: mchannel@263.net

ISBN 7-5084-3590-7

9 787508 435909

ISBN 7-5084-3590-7
定价：22.00元

21世纪高等院校规划教材

密码学——加密演算法

邓安文 编著

中国水利水电出版社

内 容 提 要

密码学的研究与应用已有几千年的历史，但作为一门科学是20世纪50年代才开始的。不可否认，互联网的广泛应用大大推动了密码学的研究与发展。大多数国家和地区都成立了密码学学会，这些学会定期召开学术会议进行学术交流，促进了密码学的研究与应用。国内外已出版了大量有关密码学的书籍，其理论研究也相对比较成熟，很多观点已达成共识。本书具有以下几个方面的特点：表述清晰、论证严谨、内容新颖、选材精良、内容丰富翔实。

本书共12章，包括：古典密码、基础数论、信息理论，对称密钥密码系统、RSA密码、非对称密钥密码系统与离散对数、数字签名、质数与大整数算术、椭圆曲线密码、公开钥基础建设、量子密码。

写一本密码学方面著作的最大困难，就是确定应包含多少数学背景知识。密码学是一个涉及广泛的学科，它需要多个数学领域的知识，包括数论、群论、环论、域论、线性代数、概率论以及信息论。同样地，熟悉计算复杂性、算法和NP完全性理论也是很有用的。在笔者看来，正是因为需要广泛的数学背景知识，所以导致学生们在开始学习密码学时感到很困难。笔者试图不使用太多的数学理论，在大多数情况下，只有需要时才引入相应的数学工具。当然，如果读者熟悉基本线性代数和模算术是会很有帮助的。另一方面，对于更专业的主题，例如信息论中熵的概念，仅给出白描似的介绍。

本书理论阐述严格完备，实例丰富，包含有大量的算法程序以及形象的图形图表，适合于读者自学，也可作为学习密码学的参考书。

本书配有免费电子教案，读者可以从中国水利水电出版社网站（www.waterpub.com.cn/softdown/）上下载电子教案及相关教学资源。

图书在版编目(CIP)数据

密码学：加密演算法／邓安文编著．—北京：中国水利水电出版社，2006
（21世纪高等院校规划教材）

ISBN 7-5084-3590-7

Ⅰ．密… Ⅱ．邓… Ⅲ．密码术—高等学校—教材 Ⅳ．TN918.1

中国版本图书馆CIP数据核字（2006）第011658号

书　　名	密码学——加密演算法
作　　者	邓安文　编著
出版　发行	中国水利水电出版社（北京市三里河路6号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787mm×1092mm　16开本　14.25印张　348千字
版　　次	2006年3月第1版　2006年3月第1次印刷
印　　数	0001—5000册
定　　价	22.00元

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

版权所有·侵权必究

序

随着计算机科学与技术的飞速发展，计算机的应用已经渗透到国民经济与人们生活的各个角落，正在日益改变着传统的人类工作方式和生活方式。在我国高等教育逐步实现大众化后，越来越多的高等院校会面向国民经济发展的第一线，为行业、企业培养各级各类高级应用型专门人才。为了大力推广计算机应用技术，更好地适应当前我国高等教育的跨跃式发展，满足我国高等院校从精英教育向大众化教育的转变，符合社会对高等院校应用型人才培养的各类要求，我们成立了“21 世纪高等院校规划教材编委会”，在明确了高等院校应用型人才培养模式、培养目标、教学内容和课程体系的框架下，组织编写了本套“21 世纪高等院校规划教材”。

众所周知，教材建设作为保证和提高教学质量的重要支柱及基础，作为体现教学内容和教学方法的知识载体，在当前培养应用型人才中的作用是显而易见的。探索和建设适应新世纪我国高等院校应用型人才培养体系需要的配套教材已经成为当前我国高等院校教学改革和教材建设工作面临的紧迫任务。因此，编委会经过大量的前期调研和策划，在广泛了解各高等院校的教学现状、市场需求，探讨课程设置、研究课程体系的基础上，组织一批具备较高的学术水平、丰富的教学经验、较强的工程实践能力的学术带头人、科研人员和主要从事该课程教学的骨干教师编写出一批有特色、适用性强的计算机类公共基础课、技术基础课、专业及应用技术课的教材以及相应的教学辅导书，以满足目前高等院校应用型人才培养的需要。本套教材消化和吸收了多年来已有的应用型人才培养的探索与实践成果，紧密结合经济全球化时代高等院校应用型人才培养工作的实际需要，努力实践，大胆创新。教材编写采用整体规划、分步实施、滚动立项的方式，分期分批地启动编写计划，编写大纲的确定以及教材风格的定位均经过编委会多次认真讨论，以确保该套教材的高质量和实用性。

教材编委会分析研究了应用型人才与研究型人才在培养目标、课程体系和内容编排上的区别，分别提出了 3 个层面上的要求：在专业基础类课程层面上，既要保持学科体系的完整性，使学生打下较为扎实的专业基础，为后续课程的学习做好铺垫，更要突出应用特色，理论联系实际，并与工程实践相结合，适当压缩过多过深的公式推导与原理性分析，兼顾考研学生的需要，以原理和公式结论的应用为突破口，注重它们的应用环境和方法；在程序设计类课程层面上，把握程序设计方法和思路，注重程序设计实践训练，引入典型的程序设计案例，将程序设计类课程的学习融入案例的研究和解决过程中，以学生实际编程解决问题的能力为突破口，注重程序设计算法的实现；在专业技术应用层面上，积极引入工程案例，以培养学生解决工程实际问题的能力为突破口，加大实践教学内容的比重，增加新技术、新知识、新工艺的内容。

本套规划教材的编写原则是：

在编写中重视基础，循序渐进，内容精炼，重点突出，融入学科方法论内容和科学理念，反映计算机技术发展要求，倡导理论联系实际和科学的思想方法，体现一级学科知识组织的层次结构。主要表现在：以计算机学科的科学体系为依托，明确目标定位，分类组织实施，兼容互补；理论与实践并重，强调理论与实践相结合，突出学科发展特点，体现

学科发展的内在规律；教材内容循序渐进，保证学术深度，减少知识重复，前后相互呼应，内容编排合理，整体结构完整；采取自顶向下设计方法，内涵发展优先，突出学科方法论，强调知识体系可扩展的原则。

本套规划教材的主要特点是：

（1）面向应用型高等院校，在保证学科体系完整的基础上不过度强调理论的深度和难度，注重应用型人才的专业技能和工程实用技术的培养。在课程体系方面打破传统的研究型人才培养体系，根据社会经济发展对行业、企业的工程技术需要，建立新的课程体系，并在教材中反映出来。

（2）教材的理论知识包括了高等院校学生必须具备的科学、工程、技术等方面的要求，知识点不要求大而全，但一定要讲透，使学生真正掌握。同时注重理论知识与实践相结合，使学生通过实践深化对理论的理解，学会并掌握理论方法的实际运用。

（3）在教材中加大能力训练部分的比重，使学生比较熟练地应用计算机知识和技术解决实际问题，既注重培养学生分析问题的能力，也注重培养学生思考问题、解决问题的能力。

（4）教材采用“任务驱动”的编写方式，以实际问题引出相关原理和概念，在讲述实例的过程中将本章的知识点融入，通过分析归纳，介绍解决工程实际问题的思想和方法，然后进行概括总结，使教材内容层次清晰，脉络分明，可读性、可操作性强。同时，引入案例教学和启发式教学方法，便于激发学习兴趣。

（5）教材在内容编排上，力求由浅入深，循序渐进，举一反三，突出重点，通俗易懂。采用模块化结构，兼顾不同层次的需求，在具体授课时可根据各校的教学计划在内容上适当加以取舍。此外还注重了配套教材的编写，如课程学习辅导、实验指导、综合实训、课程设计指导等，注重多媒体的教学方式以及配套课件的制作。

（6）大部分教材配有电子教案，以使教材向多元化、多媒体化发展，满足广大教师进行多媒体教学的需要。电子教案用 PowerPoint 制作，教师可根据授课情况任意修改。相关教案的具体情况请到中国水利水电出版社网站 www.waterpub.com.cn 下载。此外还提供相关教材中所有程序的源代码，方便教师直接切换到系统环境中教学，提高教学效果。

总之，本套规划教材凝聚了众多长期在教学、科研一线工作的教师及科研人员的教学科研经验和智慧，内容新颖，结构完整，概念清晰，深入浅出，通俗易懂，可读性、可操作性和实用性强。本套规划教材适用于应用型高等院校各专业，也可作为本科院校举办的应用技术专业的课程教材，此外还可作为职业技术学院和民办高校、成人教育的教材以及从事工程应用的技术人员的自学参考资料。

我们感谢该套规划教材的各位作者为教材的出版所做出的贡献，也感谢中国水利水电出版社为选题、立项、编审所做出的努力。我们相信，随着我国高等教育的不断发展和高校教学改革的不断深入，具有示范性并适应应用型人才培养的精品课程教材必将进一步促进我国高等院校教学质量的提高。

我们期待广大读者对本套规划教材提出宝贵意见，以便进一步修订，使该套规划教材不断完善。

21 世纪高等院校规划教材编委会
2004 年 8 月

前　言

胜利是属于“公理正义”的一方。在历史长河中，取得战争胜利的一方，往往取得历史的“解释权”，他们的意识形态就理所当然地成为所谓的“主流价值观”，因此，胜利者会被“解释”成“公理正义”的一方，而失败者会被无情地“污名化”、“妖魔化”。一般而言，胜利是属于“掌握优势资源”的一方，这些优势资源包括了军事力量、先进科技、生产技术等等，同时也包括了为人讳言的“密码技术”。

二次世界大战中，日本海军联合舰队可以说是当时世界上最强的舰队，而美国在珍珠港战役之后，海军实力已处于劣势。然而，美国却能破译日本的密码，在一连串所破译的密电中，显示出了日本海军大将山本五十六的行踪以及联合舰队的动向，使美国能够在山本五十六飞往所罗门群岛途中将其狙杀，并以劣势兵力赢得中途岛海战，这是整个太平洋战役的转折点。

勿庸置疑，密码学的确是一门实用的科学；从以往王侯将相用来对他们所发布的信息加密，到今日的电子商务、“自然人认证”、网络安全等，其中所用的核心技术就是密码学。

谈到当代密码学的核心，我们就不可避免地要了解当代密码系统的运作机制；要想对其安全性评估，就不可避免地要了解密码系统的算法；如果只是将密码系统算法轻描淡写，或只是套用一些专业术语，充其量只能是“按图索骥”，对于使用密码学技术不会有实质性的帮助。因为任何密码学所能提供的安全保证，不是建立在“入侵者无知”的假设上，其所要面对的是精通各类信息技术、了解密码系统算法的超级骇客。

诚然，信息安全的漏洞，往往不是发生在所用的密码算法上，而主要是由系统管理员造成的；也许密码系统程序员能够遵循密码学算法，编写一份近乎“完美无缺”的系统，却可能忽略了运行系统随机产生的软硬件问题；有时甚至所使用的密码学产品本身就是泄密机。即使是以当代密码机 RSA、DES、Triple DES 为加密系统的“保健 IC 卡”或是“自然人认证”，都曾发生过资料保密上的纰漏，被人视为近乎“完美无缺”的系统，却无法保证非密码层面不出问题。

本书并不想对整个信息安全的大架构进行讨论，因为除了理论上的探讨外，必须很实际地，从管理层面探讨，这并非单从算法、协定中能解释清楚的，这是大师级的工作，绝非笔者所长，但是当代密码学各类算法，有精确的数学描述方式，可以将其程序化，并将这些内容列入教材，成效会丰常显著。

一个成熟耐用的密码系统，首先要有能经得起严谨理论考验的算法。综观当代密码系统，主要可分为公开密钥密码系统（Public Key Cryptosystem）以及对称密钥密码系统（Symmetric Key Cryptosystem）两类。以前者为代表的有 RSA、ElGamal、椭圆曲线密码系统，而以后者为代表的有 DES、AES 等。这些密码系统，都要用到一些数学上的概念，而用到最多的数学相关知识，是被数学王子高斯誉为“数学女王”、被人们视为最冷门的“数

论”（Number Theory）；由于真正“了解内情”的人实在不多，所以用当代密码技术为幌子行骗的空间很大，鉴于此，笔者特将相关的基础数论内容列入本书，其实就是大学“代数”以及部分“质数”理论，这些内容都是研究当代密码学的基础。

由于当代密码技术的应用已经不是只在“纸上谈兵”而已，必须依赖程序应用。所以在本书编排上，特别用类似 C/C++/Java 的语法，对部分已成熟的算法写成伪码，只需很少的修改，就可以执行运算。对于密码算法的学习，有相当的帮助。

尽管古典密码早已不符合当代信息安全的需求，但就其中所带来的益智性乐趣，笔者实在不想只是轻描淡写；就二次大战期间德国人所用的 Enigma 密码机而言，除了历史上的乐趣外，其中的加密解密运算，其实就是对称群的置换作用，就如同转动魔方一样，是绝佳的“群”作用范例，令人着迷不已。

记得 1998、1999 年冬天，曾与 RSA 的 A 合作过的黄明德在台湾大学讲述他的一系列工作，在那时笔者领略到高深的“数论”以及“代数几何”应用到密码学的研究是深邃而引人入胜的。

本书在撰写中，费时最久的是书中的每个例题，这些例题，除了少数参考了其他文献外，大多数都是程序执行的结果整理而成，部分也取自笔者的研究内容。

另外，本书部分内容，如 RSA 密码、非对称密钥密码与离散对数、数字签名以及部分古典密码的介绍，都是笔者在清云科技大学教授“密码学”、“公开钥密码系统”时讲授过的。笔者所指导的专题学生，也分别以“RSA 电子投票研究”、“保健 IC 卡研究”、“RSA 与 PGP 研究”当作专题研究方向，专题学生林俊余与黄明宗等人，也做出以 Java 撰写的 RSA 为基础的“电子投票系统”的半成品，虽然离成熟产品还有很大距离，但也属难能可贵。感谢这些专题学生林俊余、林罔永、许丰琳、杨贺杰、黄明宗、徐效群、林克儒、陈惠甄、陈华君、陈丽君、陈俞婷、李建志、潘丹尼、吴英绮的热心参与，使得笔者在密码学的授课以及学生研究会讨论过程中，增加不少互动。所谓“教学相长”，这对本书的撰写也有一定的帮助。

在本书即将完成之际，笔者仍发现密码学实在涉略广大，许多内容只好忍痛割舍，限于各种因素，无法面面俱到，作为教材不免有遗珠之憾。

特别感谢（台湾）中央研究院数学所谢春忠教授对本书算法及算式逐一检查校阅。感谢中央研究院数学所提供笔者短期访问的机会，不少研究密码学的相关文献资料，都是在此取得。更感谢清云科技大学的系主任李振焘教授及系同事的支持与协助。另外在 Lilie 的协助下，本书也加上了“福尔摩斯密码”一节，替本书增色不少。

本书部分函数图形由数学软件 Mathematica 产生，部分精美的图案由全华科技图书绘制而成。感谢 Bletchley Park Trust 所提供的珍贵照片及 Sue May 的协助，也感谢 Brian Smith 的居中联络。

邓安文

本书的繁转简工作由李强、方春明、郝思嘉、李鑫、黄浩、王晓青、马路、王成博、王艳等人完成，在此对他们的工作表示感谢。

目　　录

序
前言
第 1 章　绪论 1
1.1　通信安全 1
1.2　公开密钥密码系统与对称密钥密码系统 5
第 2 章　古典密码 7
2.1　凯撒挪移码 8
2.2　仿射密码 9
2.3　单套字母替代法以及频率分析 10
2.4　福尔摩斯密码 13
2.5　Vigenère 密码 15
2.6　Hill 密码 20
2.7　单次密码本 21
2.8　Enigma 密码机 22
2.9　破译 Enigma 与对称群 27
第 3 章　基础数论 31
3.1　模运算与辗转相除法 31
3.2　中国余式子定理（Chinese Remainder Theorem） 36
3.3　Lagrange 定理与费马小定理 38
3.4　原根 39
3.5　二次剩余（Quadratic Residue） 41
3.6　Galois 域 45
3.7　质数理论 48
3.8　连分数 51
3.9　密码安全伪随机数生成器 54
第 4 章　信息理论 57
4.1　概率 57
4.2　完美秘密 58
4.3　熵 60
4.4　自然语言之熵 62
第 5 章　对称密钥密码系统 66
5.1　DES 与 Feistel 密码 66

5.2 Triple DES 挑战 DES......73
5.3 AES......75
5.4 IDEA......79
5.5 区块密码加密模式......83
第 6 章 RSA 密码......87
6.1 公开密钥密码系统......87
6.2 RSA 算法......89
6.3 RSA 的数论背景......92
6.4 RSA 数字签名......96
6.5 同时进行 RSA 加密和 RSA 数字签名......98
6.6 RSA-129 挑战与因数分解......100
6.7 二次筛法 Pollard 的 p－1 法......103
6.7.1 二次筛法......104
6.7.2 Pollard 的 p-1 法......107
6.8 利用 RSA 私钥因数分解......108
6.9 RSA 密码系统使用的注意事项......110
6.10 Wiener 低幂次 d 攻击......112
6.11 Rabin 密码......115
第 7 章 非对称密钥密码系统与离散对数......119
7.1 Pohlig-Hellman 密码与离散对数......120
7.2 Diffie-Hellman 密钥交换......123
7.3 ElGamal 密码......126
7.4 Pohlig-Hellman 算法......127
7.5 Index Calculus......129
第 8 章 数字签名......131
8.1 数字签名方案......131
8.2 RSA 盲签名......133
8.3 Hash 函数简介......135
8.4 生日攻击......136
8.5 ElGamal 数字签名......137
8.6 DSA 数字签名......140
8.7 Schnorr 数字签名......143
8.8 Nyberg-Rueppel 数字签名......144
8.9 MD5 Hash 函数......147
8.10 SHA-1 Hash 函数......150
8.11 信息校验码 MAC......152

第 9 章　质数与大整数算术……154
9.1　大整数的加减乘法……154
9.2　大整数的除法……157
9.3　Montgomery 算术……159
9.4　Miller-Rabin 质数测试……161
9.5　Agrawal-Kayal-Saxena 算法……163
9.6　公开密钥密码的质数……165
9.6.1　强质数……165
9.6.2　DSA 质数……166
9.7　Java 的 BigInteger Class……167
9.8　大整数算术与数论套件及软件……171
第 10 章　椭圆曲线密码……173
10.1　椭圆曲线……174
10.2　椭圆曲线（mod p）……179
10.3　加权投影坐标……183
10.4　定义在 Galois 域 $\mathbb{F}_{2^m}$ 的椭圆曲线……185
10.5　密码安全曲线……188
10.6　将信息转化为椭圆曲线代码……189
10.7　椭圆曲线公开密钥密码算法……190
10.8　椭圆曲线因数分解……196
10.9　ECCp-109 挑战……199
10.10　并行 Pollard Rho 法……201
第 11 章　公开密钥基础建设……204
11.1　认证机构 CA……204
11.2　X.509……206
11.3　认证机构 CA……207
第 12 章　量子密码……208
12.1　量子实验……208
12.2　量子密钥分配……210
12.3　浅谈 Shor 之量子算法……212
参考文献……214

第 1 章　绪论

密码学（Cryptology，Cryptography）是指秘密书写、加密信息、隐藏信息内容的科学，同时也泛指与密码有关的科学。人类文明史本是尔虞我诈、纷争不断，自从发展大规模战争以来，密码的使用层出不穷。如罗马共和国时代的执政官凯撒（GaiusJuluis Caesar，公元前 100 年到公元前 44 年），就多次使用一种字母替代密码，称为恺撒挪移码（Caesar Shift Cipher）。二次世界大战中，因为盟军及时有效地破译了德国的 Enigma 密码机，至少挽救了百万生灵免受涂炭。二战后英国接收了为数众多的 Enigma 密码机当作战利品，多数卖给了英联邦成员国，这些英联邦成员国都相信这些德国货真能保密，这使得 Enigma 密码机成了为英国提供各国机密的一个渠道。

密码学的应用已经不再局限于“军国大事”了，虽然密码学的相关研究仍以负责“国家安全”的研究单位为主，这些研究单位拥有大量的研发人力资源（至少西方国家如此）。与大多数学科不同的是，密码学的许多研究成果是不公开的、也不希望被外界披露，密码技术的输出更受到严格管制，这也使得密码学的研究蒙上了神秘色彩。随着近十年 Internet 网络的发展，电子商务的开展、IC 卡的使用，都必须依赖可靠的通信安全，以防止怀有恶意的第三者——如骇客（Hacker）入侵，而密码学也因此成为大众科学。加上近来政府大力推广密码学技术，如电子公文交换系统、保健 IC 卡、身份证甚至将来可能面临使用的电子投票等；业界更是大力推销密码产品，好像不买不安全，买了就有了保障，这些密码学技术多半来自于所谓的“密码技术先进的国家”，大多数使用者不了解也无从检验其安全性。这些密码技术、密码商品是否真如推销者所言，还是另有玄机？

1995 年，德国的明镜杂志（Der Spiegel）以及美国的 Baltimore Sun 都报道了位于瑞士的一家著名密码公司 Crypto AG 在他们的部分密码商品上加了后门，并提供美国国家安全局（National Security Agency，NSA）后门细节，客户资料可由此后门窃取，这件事在当时举世哗然。

密码学的应用也早已超越单纯的信息安全的领域，逐渐渗透到我们的日常生活当中。可以确定的是，我们逐渐生活在一个密码技术的年代，希望这些密码技术真正能保障各种隐私，而不是以密码技术为幌子的加上后门的“特洛伊木马”来服务某些人，否则电影“全民公敌”的故事情节将会以密码技术为包装重演。

1.1　通信安全

密码学的核心就是密码加密方法，而每种密码法都可视为某种加密法，即算法（Algorithm）加上密钥（Key）的组合。加密解密的场景（Scenerio）如图 1-1 所示。

- Alice 为传递信息者即加密者，将明文（Plaintext）加密成密文（Ciphertext）。
- Bob 为接收信息者即解密者，将密文解密成明文。
- Eve 为敌对的第三者，它在传递信息的过程中截收密文。

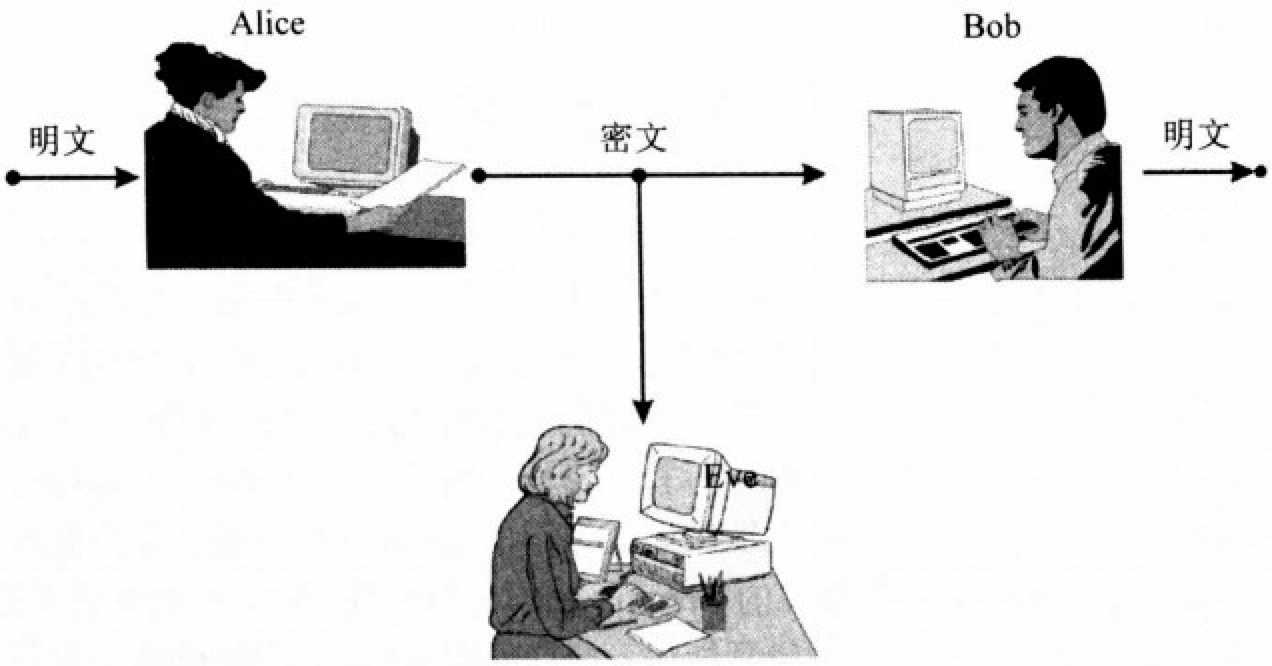


图 1-1　通信场景

Eve 可能会有以下目的：

（1）阅读密文。

（2）找出密钥，解读密文。

（3）篡改或修改 Alice 传给 Bob 的信息。

（4）假扮成 Alice，与 Bob 传递信息，让 Bob 误认为他还是与 Alice 通信。

若 Eve 只是阅读密文，似乎是无害的，而上述其他可能性的发生都来自于 Eve 的意愿和他分析密码的技术能力问题。可以依据 Eve 对信方式和密码系统掌握的程度，将攻击方式分为如下几类：

（1）破译密文层次的攻击方式。

- 密文攻击（Ciphertext-Only Attack）：Eve 只知道密文，尝试恢复成相应明文，或者找出密钥。
- 明文攻击（Known-Plaintext Attack）：Eve 知道至少一组明文/密文对照文，尝试找出密钥，或者尝试破译解读其他密文。
- 选择明文攻击（Chosen-Plaintext Attack）：Eve 能够将明文编译成密文，但不知道密钥，尝试找出密钥，或者破译解读其他密文。
- 选择密文攻击（Chosen-Ciphertext Attack）：Eve 能够将密文破译成明文，但不知道密钥，尝试找出密钥。

（2）主动式攻击（Active Attack）方式。这一大类的攻击方式有时也需要先进的硬件设备，也包括篡改信息、假扮传讯者的攻击方式，广义地说，也包括即时攻击（Timing Attack）和贿赂对方人员等。比如所谓的暴风雨攻击（Tempest Attack），就是指在 Alice 完成密文编译之前，Eve 能侦测出 Alice 电脑输入的明文的每一字母所发出的特有电磁波信息。

例 1.1　在第二次世界大战时，德国使用 Enigma 密码机，同盟国破译的攻击方式属于明文攻击。最常出现的例子就是，德国每天的气象报告的密电，将德文的气象信息，在密电中编译成密文，如此只要找出足够多明文/密文对照文，加以分析后就能破译出当日德国所使用的密钥。

例 1.2　在密文攻击中，若采取对所有可能的密钥进行检试，这种方法称为穷举法攻击

（Exhaustive Attack），或称为暴力攻击（Brute-Force Attack）。比如说密码 DES 的密钥有效长度为 56bit，一次所有可能的密钥数为 2^{56}，暴力攻击成功的可能性还是有的。

在现代密码学中，针对信息保密有一个非常重要的假设，即 Kerckhoffs 原理。

Kerckhoffs 原理：应该假设敌人已经知道所使用的密码系统的保密方法。

因此密码系统的使用，其安全性在于密钥的保密，而假设破译者早已知道算法。今天，算法应广义地包括密码系统的加密、解密算法、所用的密码元件、所使用的协议、随机数生成器、软件硬件设备。而密码系统应对信息安全提供以下四大功能：

（1）机密性（Confidentiality）：Eve 根本无法破译解读 Alice 传给 Bob 的密文，而其主要工具就是该密码系统的加密、解密算法。

（2）数据完整性（Data Integrity）：Bob 想要确定 Alice 所传递的信息未被篡改，但在传递上也可能会出错，此时就应该考虑具有侦错功能的编码理论（Coding Theory），而密码杂凑函数，提供了可侦测数据是否遭篡改的方法。

（3）可认证性（Authentication）：Bob 希望能确定传递的信息“应该”是由 Alice 本人所发送的，而非他人所伪造。

（4）不可否认性（Non-Repudiation）：Alice 不能否认是她所发送的信息。

可认证性与不可否认性似乎是相近的概念，但相互之间仍有区分：以传统的对称密钥密码（Symmetric Key Cryptosystems）为例，Alice 与 Bob 共同使用一把相同密钥对此对称密钥密码加密解密，在 Bob 收到密文之际，就假设除了 Bob 之外只有 Alice 知道密钥，因此密文应该是 Alice 传送的，而非他人伪造，因此可认证性自然成立；但在有争议时，Alice 否认是她发送这份信息的，因为不排除 Bob 也将密钥告诉了第三者，或伪造密文，两方各执一辞，因此不可否认性在此不可能实现。此时，不可否认性的解决方案，必须借助于公开密钥密码系统（Public Key Cryptosystem），如数字签名（Digital Signature）。不可否认性排除了 Alice 寻找不在场证明（Alibi）的可能性，这为电子商务，特别是网络交易提供了重要的安全功能。

如何界定一密码系统的安全程度？我们先讨论以下术语。

定义 1.1 一个密码系统为无条件安全（Unconditionally Secure）就是指即使接收到无限密文，也无法确定其密钥。

定义 1.2 一个密码系统为计算上安全（Computationally Secure）就是指该密码系统满足破解密文的花费远远大于所加密信息的价值，且破解密文所花费的时间远远多于该信息的有效时间。

定义 1.3 一个密码系统为可证明安全（Provable Secure）就是指该密码安全性问题可转化成某个研究人员公认的困难问题。

事实上，一个密码是无条件安全的，只有在密文的长度与密钥的长度大致相同时，才会成立，在实际上，只有单次密码本（One-Time Pad）才是无条件安全的，其他的都不是。而公开密钥密码系统 RSA，是可证明安全的，因为该密码系统的安全性问题，在大量的研究下，一般可转化成质因数解分的问题，而质因数分解的问题，一般认为是很困难的。事实上，能将密码的安全程度量化，关键就在计算上安全这个概念；这与度量某一个算法的时间与存储空间的计算复杂度（Computational Complexity）有直接关联。

例 1.3 函数 $4n^2+7n+12$ 的数量级为 n^2，通常用大写“O”或小写“o”表示为

$$4n^2+7n+12=O(n^2) \text{ 或者 } 4n^2+7n+12=o(n^3)$$

定义 1.4 令 f、g 为 n 的函数，

$$f(n)=O(g(n)) \Leftrightarrow \lim_{n\to\infty}\frac{|f(n)|}{|g(n)|}<c$$

对某正数c成立，则

$$f(n)=o(g(n)) \Leftrightarrow \lim_{n\to\infty}\frac{|f(n)|}{|g(n)|}=0。$$

另外，信息时代常用的单位 MIPS（One-Million-Instruction-Per-Second，1 秒执行一百万次命令），20 年前的计算机 VAX-11/780 就大约是 1 MIPS 计算机，1 MIPS 计算机在一年约可执行 3×10^{13} 次指令，而 100 MHz Pentium 计算机是 50MIPS 计算机，运行大约是 VAX-11/780 的 50 倍，而 1GHz Pentium 计算机的运行速度 500MIPS。单位“MIPS 年”是指 1 MIPS 电脑在一年内所执行指令的次数，也可看作时间单位。表 1-1 所示为不同数量级算法处理 $n=10^6$ 在 1MIPS 电脑所需的时间。

表 1-1 算法处理 $n=10^6$ 在 1MIPS 电脑所需的时间

算法	计算复杂度	所需运算次数	1MIPS 电脑执行时间
常数	O(l)	1	1 μsec
线性	O(n)	10^6	1 sec
平方	$O(n^2)$	10^{12}	11.6 天
立方	$O(n^3)$	10^{18}	32000 年
指数	$O(2^n)$	10^{301030}	10^{301017}

例 1.4 RSA 密码系统是可证明安全的，该密码系统可转化成质因数分解的问题，以目前的代数域筛法（Algebraic Number Field Sieve Method）算法尝试破解，对于不同的 RSA 密钥长度，其运算次数如表 1-2 所示。

表 1-2 使用代数域筛法分解不同密钥长度的 RSA 模数的运算次数

RSA 密钥位数	MIP 年
512	8400
768	5×10^7
1024	6×10^{10}
2048	7×10^{19}
3072	3×10^{26}
4096	6×10^{31}

注意：许多对称密钥密码以及所有的公开密钥密码系统皆可在 NP 时间内破解。注[1]

注[1] NP（Nondeterministc Polynomial）问题是指在 Nondeterministc Turing 机上所需多项式时间方可破解，即 $O(t^m)$时间 $(m\in R)$。而 Nondeterministc Turing 机器只指一有限状态机可作无限多次读写动作，且可平行尝试所有的猜想，并需多项式时间可破解，而与现在计算机较为相近的是 Nondeterministc Turing 机，有关此问题，可参阅：Daniel I.A.Cohen，“*Introduction Theory*”，Second Edition，John Wiley &Sons，Inc.（1997）

1.2 公开密钥密码系统与对称密钥密码系统

定义 1.5 考虑 Alice 传信息给 Bob，如图 1-2 所示。

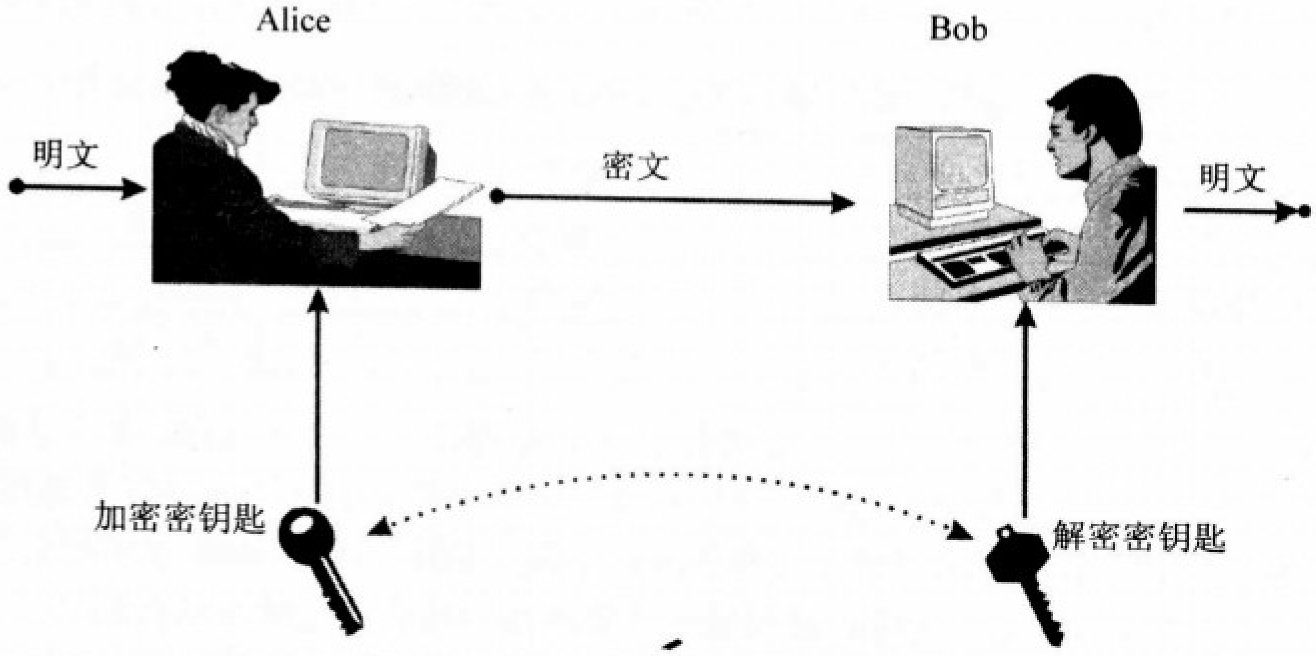


图 1-2 对称密钥/公开密钥密码系统

- 一密码系统为对称密钥密码系统（Symmetric Key Cryptosystem）
  ⇔ 其加密密钥=解密密钥，钥匙是保密的。
- 一密码系统为非对称密钥密码系统（Asymmetric Key Cryptosystem）
  ⇔ 其加密密钥 ≠ 解密密钥
- 一密码系统为公开密钥密码系统（Public Key Cryptosystem）
  ⇔ 其加密密钥 ≠ 解密密钥，且加密密钥为公钥（Public Key）而解密密钥为私钥（Private Key）

由此定义可知，所有的公开密钥密码系统都是非对称密钥密码系统，与对称密钥密码系统没有交集；另有一种称为 Pohlig-Hellman 的密码系统，它是非对称密钥密码系统，但不是公开密钥密码系统，它的加密密钥及解密密钥皆保密的。对称密钥密码系统包含所有传统的加密系统，如古代的凯撒挪移码、Viginere 码、单次加密簿以及二次大战所采用的 Enigma 密码机和战后计算机发明后所广为采用的 DES，及新的加密法 AES、IDEA 等。而公开密钥密码系统则有 RSA 密码、将 Diffie-Hellman 密钥交换加以推广的 Elgamal 密码、椭圆曲线密码（Elliptic Curve Cryptosystems）以及大多数的数字签名（Digital Signature）算法。

对称密钥密码系统与公开密钥密码系统在实际应用上有什么优劣？在不同的要求上有何特性？

（1）对称密钥密码系统一般而言，其运算的速度远远快于公开密钥密码系统，这在大批数据加密上有其实用价值，如 DES 的加密解密速度就比 RSA 密码系统快千倍以上。

（2）公开密钥密码系统的密钥长度可根据需求而增长，不像对称密钥密码系统的密钥长度是固定的，如对称密钥密码系统 DES 的密钥为固定的 56bit，AES 的密钥为固定的 128bit，192bit 和 256bit 三种，而公开密钥密系统 RSA 的密钥为 512bit，甚至可依据需求而增长，从而增加计算安全度，故拥有长密钥的 RSA，其计算安全度要远远高于 DES。通常，对称密钥

密码系统是以暴力攻击估计其计算安全度，所以它的计算复杂程度为指数函数 $O(2^n)$，其中对称密钥密码系统的密钥为 nbit；而 nbit 密钥的公开密钥密码系统 RSA 是以代数域筛法攻击估计其计算安全度的，其计算复杂度为亚指数（Subexponential）函数

$$e^{(1.923+o(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}} = e^{O(e^{1/3}(\ln e)^{2/3})}$$

不同长度密钥的 RAS 与对称密钥密码系统所提供的相同的计算安全度如表 1-3 所示。

表 1-3 不同长度密钥的 RSA 与对称密钥密码系统所提供的相同的计算安全度

对称密钥密码	56bit	80bit	112bit
RSA	512bit	1024bit	2048bit
对称密钥密码	128bit	192bit	256bit
RSA	3072bit	7680bit	15360bit

（3）在公开密钥密码系统中，数字签名是可以实现的，以 RSA 为例，其方法如下：Alice 要对一个文件数字签名给 Bob，她先用私钥对要签名的文件加密，因为任何人都可取得 Alice 的公开密钥，即加密密钥，加密函数与解密函数互为反函数，因此 Bob 就可用公开密钥对已加密的 Alice 数字签名解密，这样，通信安全所要求的不可否认性就可以做到了。

（4）在实际应用中，所使用的加密软件大都是将对称密钥密码系统与公开密钥密码系统整合而成，如 PGP 电子邮件加密软件的早期版本注[2]，就使用了公开密钥密码 RSA 以及对称密钥密码系统 IDEA，分别在加密的速度上以及通信安全加以考虑，以 RSA 加密传送 IDEA 的密钥，而内文加密采用 IDEA，这样，加密速度与通信安全性得以兼顾，并能解决分配密钥的问题。

公开密钥密码系统的发展是比较短的，最早所思考的问题，是对传统对称密钥密码系统的密钥分配问题。考虑在一庞大的机构中，任何两人都可彼此利用对称密码加密解密信息，这导致密钥的总数相当庞大且不易管理；而公开密钥密码系统的概念大大简化了密钥的管理问题，在 1976 年 Diffie 与 Hellman 就提出一篇划时代的文章注[3]，其中提出公开密钥密码的概念，如单向陷门函数（Trapdoor One-way Function）等；而在 1977 年由 Rivest、Shamir、Adleman 所提出的一种加密法，即后来的 RSA 密码，在密码学的发展上起了根本的变化，公开密钥密码系统终于产生了。

注[2] 读者可以到http://www.pgpi.org 下载 PGP电子邮件加密软件。

注[3] Diffie and Hellman，"New directions in cryptography." IEEE Trans.Inform Theory IT-22 P.644654,1976。

第 2 章　古典密码

近代密码技术必须在计算机上运行，而人类使用密码却已经有几千年的历史，使用密码的目的，就是不让敌方或不该知道信息内容的人知道信息内容。对使用汉字的中国人而言，很早就有妇女们所使用的“女书”，对于不该知道内容的男人而言，这就是一种密码技术。而西方使用字母拼音文字的民族，早期的密码技术主要以字母的代替（Substitution）以及字母的位移（Transposition）为主，有时也混合代码法代替整个单字或词组。

在中世纪的伊斯兰国家，学者致力于研究可兰经，甚至分析了经文每个不同字词及字母的出现频率；当时的伊斯兰国家正处于高度文明的时期，数学与语言学都处于很高水平，这也为密码分析学（Cryptanalysis）提供了可能的环境。在欧洲还处于黑暗时期时，远在中东与近东的回教徒早已熟悉用（Frequency Analysis）频率分析破译的单套字母代替密码；简单的单套字母替代密码在频率分析未发明之前如同无字天书，但在频率分析破译法产生后破译起来就易如反掌了。

16 世纪，一种名为 Vigenere 密码的多套字母替代密码诞生了，这在当时被视为无法破译的密码，一直到了 19 世纪才被破译，而破译的方法仍是以频率分析为主。

人类在这个时期，破译密码的技术要高于编译密码的技术。欧洲列强矛盾冲突引发了的第一次世界大战，密码学家此时无法提供高明的密码编译方法，只不过是将多套字母替代与位移结合产生密码，如 Playfair 或 ADFGVX 被勉强使用，除了刚出炉之际还可以应付一下，接下来就是等着对方密码分析师应用频率分析法破译。如此应急式的密码，再加上在当时防御武器优于进攻武器的情况下，任何“早已被破译的”进攻战计划岂有不败之理。

密码技术在此时的发展，实在令人沮丧，但一种真正无法破译的密码单次密码本（One-Time Pad）在战争末期发明了，值得一提的是，冷战时期，美苏之间领导人的热线就使用这种方式加密。然而此类密码法所需成本极高，每加密一次就要用不同的密钥，这在军事应用上尤其困难，故无法大量使用。

古典密码发展的最后一个阶段，应是二次大战前后所使用的滚轮（Rotor Machine）编码，最著名的应属德国所采用的 Enigma 密码机，拥有如同天文数字的密钥数量，这是无法用传统的频率分析法破译的，然而在德国发动战争之前，被波兰密码分析师破译成功，随着战争爆发，德国继续加强他们的密码机，这项破译技术也适时地转移到英国，Bletchley Park 的密码分析师继续试图破译德国的密码机，借助他们所制造的机器，对所截收密码的分析结果进行计算对比，而这种用来协助破译密码的机器 Colossus，就是计算机的前身。由于密码分析的需求，人们发明了计算机来对付机械密码机，而在计算机发明之后，人们又利用它发展更新、更强的当代密码技术。

在本章中，我们将讨论古典密码及其破译技术，所讨论的纲要如下：

- 凯撒挪移码。
- 仿射密码。
- 单套字母替代法与频率分析。

- 福尔摩斯与密码。
- Vigenere 码与频率分析。
- Hill 码。
- 单次密码本。
- Enigma 密码机。
- 破译 Enigma 密码与对称群。

2.1 凯撒挪移码

凯撒挪移码（Caesar Shift Cipher）是一种简易的单套字母替代法，即将明文字母（Plain Alphabet）改成在它后三位的字母，若为 x、y、z，则换为 A、B、C，成为密文字母（Cipher Alphabet）。比如撒名言 Veni vidi vici（我来了，我见了，我征服了）就可以加密成 YHQL YLGL YLFL；而解密程序就是把密文字母改成在它前三位的字母，如为 A、B、C，则换为 x、y、z。

为计算方便起见，可将字母化为数字代码，如图 2-1 所示为字母数字代码。

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

图 2-1 字母数字代码

凯撒挪移码也不限定将明文字母只挪移后三位，下例中说明了其加密解密过程。

例 2.1 Alice 要将明文“gaul is divided into three part”加密成密文，传给 Bob。

- （密钥产生）Alice 与 Bob 协定编码方式为明文字母后移 4 位，即加密密钥及解密密钥同为 K=4。
- （加密）Alice 将明文“gaul is divided into three part”转为数字代码：（6，0，20，11，8，18，3，8，21，8，3，4，3，8，13，19，14，19，7，17，4，4，15，0，17，19）。使用加密函数 E（m）≡ m+k=m+4（mod 26）计算得：（10，4，24，15，12，22，7，12，25，12，7，8，7，12，17，23，17，23，11，21，8，8，19，4，21，23），即密文“K，E，Y，P，M，Z，M，H，I，H，M，R，X，R，X，L，V，I，I，T，E，V，X”。
- （解密）Bob 收到密文“KEYPMZMHIHMRXRXLVIITEVX”=（10，4，24，15，12，22，7，12，25，12，7，8，7，12，17，23，17，23，11，21，8，8，19，4，21，23）
  使用加密函数 D（c）≡c-k=c-4（mod 26）计算，并考虑空格，可还原明文：（6，0，20，11，8，18，3，8，21，8，3，4，3，8，13，19，14，19，7，17，4，4，15，0，17，19）=“gaul is divided into three part”。

例 2.2 Eve 截收到由 Alice 传给 Bob 的一份密文，并知道是以凯撒挪移码编码。密文为

“penpxre”，Eve 用暴力攻击，使用所有 25 把钥匙，即解密函数 D_k（c）≡c-k (mod 26) (k=1、2、…25)，得

（k=1）‘odmowqd’　（k=2）‘nclnvpc’　（k=3）‘mbkmuob’　（k=4）‘lajltna’
（k=5）‘kziksmz’　（k=6）‘jyhjrly’　（k=7）‘ixgiqkx’　（k=8）‘hwfhpjw’
（k=9）‘gvegoiv’　（k=10）‘fudfnhu’　（k=11）‘etcemgt’　（k=12）‘dabdlfs’
（k=13）‘cracker’　（k=14）‘bqzbjdq’　（k=15）‘apyaicp’　（k=16）‘zoxzhbo’
（k=17）‘ynwygan’　（k=18）‘xmvxfzm’　（k=19）‘wluweyl’　（k=20）‘vktvdxk’
（k=21）‘ujsucwj’　（k=22）‘tirtbvi’　（k=23）‘shqsauh’　（k=24）‘rgprztg’
（k=25）‘qfoqysf’

这样 Eve 就会发现在 k=13 才得到有意义的文字“cracker”。

2.2　仿射密码

与凯撒挪移码一样，仿射密码（Affine Cipher）也是一种单套字母替代法，以拉丁字母 26 个字母为例，将字母化为数字代码 a=0、b=1、…、z=25，其加密函数为

$$E(m) = \alpha x + \beta (\mathrm{mod}\ 26)$$

其中 α 、β 为整数且 α 必须与 26 互质。为何 α 一定要与 26 互质？假设 $\gcd(\alpha, 26) = g > 1$，此时 E(.) 将不是 1-1，不同明文字母 x_1、x_2 将会对应到相同值；以 $\gcd(\alpha, 26) = g > 1$ 而言，g 只可能有 g=2 或 g=13，以 g=2 时为例，只需 $x_1 \equiv x_2 (\mathrm{mod}\ 13)$，而当 g=13 时，只需 $x_1 \equiv x_2 (\mathrm{mod}\ 2)$ 时都会

$$\alpha x_1 + \beta \equiv \alpha x_2 + \beta (\mathrm{mod}\ 26)\text{。}$$

当加密函数不是 1-1 时，不存在反函数，这样的情况必须排除，以集合

$$\mathbb{Z}/26 = \{0, 1, 2, 3, 4, \cdots, 25\}$$

而言，可以在该集合上进行模加法、模减法以及模乘法，但是“模除法”就不是每个元素能做的（具体内容参考本书 3.1 节），此时要考虑对于 $x \in \mathbb{Z}/26$，如何判别 x 有“乘法反元素”？就是是否存在 $y \in \mathbb{Z}/26$，使得

$$xy \equiv 1 \ (\mathrm{mod}\ 26)\text{，}$$

以及如何求解 y？要判断 x 是否有“乘法反元素”，只需观察 x 是否与模数 n 互质，而求解 y 的方法，较为普遍的方式就是使用广义辗转相除法（Extended Euclidean Algorithm）（具体内容参考本书 3.1 节）。

例 2.3　集合 $\mathbb{Z}/26$ 中各元素 x 的“乘法反元素” x^{-1} 对照表如下所示（当 gcd(x,26)>1 时，x 无“乘法反元素”，故排除在对照表之外）：

x	1	3	5	7	9	11	15	17	19	21	23	25
x^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

有了 $\mathbb{Z}/26$ 乘法反元素对照表，就可以很容易找出加密函数 $E(m) = \alpha m + \beta$ 的反函数，即解密函数

$$D(c) = \alpha^{-1}(c - \beta)\text{。}$$

例 2.4 Alice 欲将明文 m=“affine”用仿射密码加密，传讯给 Bob，Bob 来解读。

- （密钥产生）Alice 与 Bob 事先协定一把密钥 K=(3,8)其中 gcd(3,26)=1。
- （加密）Alice 使用加密函数 $E(m) \equiv 3m+8 \pmod{26}$ 计算得

$$\begin{aligned} m &= \text{'affine'} \\ &= (0,5,5,8,13,4) \xrightarrow{E(.)} (8,23,23,6,21,20) \\ &= \text{'IXXGVU'} = c \end{aligned}$$

- （解密）Bob 收到密文 c='IXXGVU'使用解密函数

$D(c) \equiv 3^{-1}(c-8) \equiv 9(c-8) \pmod{26}$ 计算得

$$\begin{aligned} c &= \text{'IXXGVU'} \\ &= (8,23,23,6,21,20) \xrightarrow{D(.)} (0,5,5,8,13,4) \\ &= \text{'affine'} = m \end{aligned}$$

2.3 单套字母替代法以及频率分析

前面所讲的凯撒挪移码以及仿射密码均为单套字母替代法的特例，一般来说以英文 26 个字母为例，所谓单套替代法，就是将明文字母用其他的密文字母代替，进行加密，而解密过程就是加密运算的逆运算。更精确地说，就是考虑字母集合

$$A=\{a,b,c,d,\cdots,x,y,z\}$$

而加密函数就是 1-1 函数 $f: A \to A$，而解密函数就是 f^{-1}，为 f 的反函数。

定义 2.1 （置换，Permutation）：

令

$$A=\{1, 2, 3, \cdots, n\}$$

为有限集合，

令

$$f: A \to A$$

为函数，称 f 为 A 上的置换（Permutation）⇔ f 为 1-1。

以 S_n 代表 A 上所有置换成的集合，即

$$S_n = \{ f: A \to A \mid f \text{ 为 A 上的置换} \},$$

一般称为对称群（Symmetric Group）注[1]。

例 2.5 令 A={1，2，3，4，5}，f 为 A 上的置换使得 f(1)=2，f(2)=4，f(4)=5，f(5)=1，f(3)=3。可将置换 f 表达为

$$f = \begin{pmatrix} 12345 \\ 24351 \end{pmatrix}$$

或者

$$f = (1245),$$

第二种表达方式中的（1 2 4 5）即表示

注[1] 可参考本书第三章群的内容

$$1\mapsto 2, 2\mapsto 4, 4\mapsto 5, 5\mapsto 1,$$

如此对应关系形成循环，而 3 不随置换改变，故未出现；一般称之为 Cycle。

在此处，我们会以第一种方式表达置换（当然第二种方式是较为经济的，且较具有代数意义)。

例 2.6　Alice 想以单套字母替代法加密与 Bob 传递信息，事先约定密钥为一串字母

KEYWORD,

此时加密函数为

$$f=\begin{pmatrix}\text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}\\ \text{KEYWORDABCFGHIJLMNPQSTUVXZ}\end{pmatrix},$$

其中前面 7 个明文字母分别对应密文字母为：$a\mapsto K$，$b\mapsto E$，$c\mapsto Y$，$d\mapsto W$，$e\mapsto O$，$f\mapsto R$，$g\mapsto D$，剩下的就按照未用的字母顺序对应。而解密函数 f^{-1} 就是将加密函数 f 的表达式上下两行交换，并重新按顺序排列：

$$f^{-1}=\begin{pmatrix}\text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}\\ \text{HIJGBKLMNOAPQRESTFUVWXDYCZ}\end{pmatrix}。$$

假设明文为　　m= “monoalphabeticsubstitutioncipher”,

则密文为　　c= “HJIJKGLAKEOQBYPSEPQBQSQBJIYBLAON”。

以 26 个字母置换的总数共有

$$26!=403291461126605635584000000>2^{56}\approx 4\times 10^{26},$$

若以暴力攻击法逐一进行置换运算，代入检查，这简直是天文数字，要比近代密码 DES 的密钥数还多，即使用电脑逐一检查也有困难，但是明文所用的语言若为英文，其 26 个字母所出现的频率并不是平均的，因此可由密文所出现字母的频率，与常用英文字母出现的频率比率作比较，只要接收到的密文够长，应足以破译，从如表 2-2 所示的英文字母出现频率比率可知，其中最常出现的 3 个字母分别为：e、t、a，而最少出现的字母为 q、y、z。

表 2-2　英文字母出现频率

字母	百分比	字母	百分比
a	8.2	n	6.8
b	1.5	o	7.5
c	2.8	p	1.9
d	4.2	q	0.1
e	12.7	r	6.0
f	2.2	s	6.3
g	2.0	t	9.0
h	6.1	u	2.8
i	7.0	v	1.0
j	0.1	w	2.4
k	0.8	x	2.0
l	4.0	y	0.1
m	2.4	z	0.1

另外最常出现的双字母组合为：

th(3.15%)，he(2.51%)，an(1.72)，in(1.69%)，er(1.54%)，re(1.48%)，es(1.45%)，on(1.45%)，ea(1.31%)，ti(1.28)，at(1.24%)，st(a.21%)，en(1.20%)，nd(1.18%)等。

最常出现的三字母组合（Trigram）为：

the，ing，and，her，ere，ent，tha，…。

运用这些资料，明文为英文的单套字母加密的密文，就不是那么牢不可破了。至于其他文字，如德文、法文都可以用单套字母替代法，破译的方法可以像英文一样用频率分析。

例 2.7 有密文如下注[2]：

BT JPX RMLXPCUV AMLXICBJP IBTWXVR CI MLMT'R PMTN,MTN YVCJX CDXV MWMBTRJ JPX AMTNGXRJBAH UQCT JPX QGMRJV CI JPX YMGG CI JPX HBTW'R QMGMAX, MTN JPX HBTW RMY JPX QMVJ CI JPX PMTN JPMJ YVCJX.JPXT JPX HBTW'R ACUTJXTMTAX YMR APMTWXN, MTN PBR JPCUWPJR JVCUFGXN PBL,RC JPMJ JPX SCBTJR CI PBR GCBTR YXVX GCCRXN, MTN PBR HTXXR RLCJX CTX MWMBTRJ MTCJPXV, JPX HBTW RQMHX,MTN RMBN JC JPX YBRX LXTCI FMFEGCT, YPCRCXDXV RPMGG VXMN JPBR YVBJBTW,MTN RPCY LX JPX BTJXVQXJMJBCT JPXVXCI, RPMGG FX AGCJPXN YBJP RAMVGXJ, MTN PMDX M APMBT CI WCGN MFCUJ PBR TXAH, MTN RPMGC FX JPX JPBVN VUGXV BT JPX HBTWNCL.JPXT AMLX BT MGG JPX HBTW'S YBRX LXT;FUJ JPXE ACUGN TCJ VXMN JPX YVBJBTW,TCV LMHX HTCYT JC JPX HBTW JPX BTJXVQVXJMJBCT JPXVXCI ,JPXT YMR HBTW FXGRPMOOMV WVXMJGE JVCUFGXN, MTN PBR ACUTJXTMTAX YMR APMTWXN BT PBL, MTN PBR GCVNR YXVX MRJCTBRPXN. TCY JPX KUXXT, FE VXMRCT CI JPX YCVNR CI JPX HBTW MTN PBR GCVNR, AMLX BTJC JPX FMTKUXJ PCURX;MTN JPX KUXXT RQMHX MTN RMBN, C HBTW, GBDX ICVXDXV;GXJ TCJ JPE JPCUWPJR JVCUFGX JPXX, TCV GXJ JPE ACUTJXTMTAX FX APMTWXN;JPXVX BR M LMT BR JPE HBTWNCL, BT JPX NMER CI JPE IMJPXV GBWPJ MTN UTNXVRJMTNBTW MTN YBRBCL,GBHX JPX YBRNCL CI JPX WCNR.YMR ICUTN BT PBL;YPCL JPX HBTW TXFUAPMNTXOOMV JPE IMJPXV, JPX HBTW, B RME, JPE IMJPXV, LMNX LMRJXV CI JPX LMWBABMTR, MRJVCGCWXVR,APMGNXMTR, MTN RCCJPRMEXVR, MRJVCGCWXVR, APMGNXMTR, MTN RCCJPRMEXVR; ICVMRLUAP MR MT XZAXGGXTJ RQBVBJ, MTN HTCYGXNWX, MTN UTNXVRJMTNBTW, BTXVQVXJBTW CI NVXMLR, MTN RPCYBTW CI PMVV RXTJXTAXR,MTN NBRRCGDBTW CI NCUFJR, YXVX ICUTN BT JPX RMLX NMTBXG, YPCL JPX HBTW TMLXN FXGJXRPMOOMV; TCY GXJNMTBXG FX AMGGXN ,MTN PX YBGG RPCY JPX BTJXVQVXJMJBCT.JPX IBVTJ ACNXYXVN BR CJPXGGC.

在此密文中，“JPX”这个字母出现的次数很多，因此可能为英文三字母组合最常出现的

注[2] 该段摘自《码书》（Simon Singh 的“The Code Book”）密码挑战——十阶通达一万英镑的第一个密文。

"the"，然后再套用到其他密码文中出现 J、P、X 的词中，再加上频率分析的方法，这样很快就能解出这段密文的明文如下：

In the same hour came forth fingers of a man's hand,and wrote over against the candlestick upon the plaster of the wall of the king's palace;and the king saw the part of the hand that wrote. Then the king's countenance was changed, loins were loosed, and hes knees smote one against another. The king cried aloud to bring in the astrologers, the chaldeans, and the soothsayers, And the king spake, and said to the wise men of Babylon, whosoever shall read this writing, and show me the interpretation thereof, shall be clithed with scarlet, and have a chain of gold about his neck, and have a chain of gold about his neck, and shall be the third ruler in the kingdom. Then came in all the king's wise men; but they could not read the writing, nor make known to the king the interpretation thereof. Then was king Belshazzar greatly troubled, and hes countenance was changed in him, and his lords were astonished. Now the queen, by reason of the words of the king and his lords, came into the banquet house; and the queen spake and said, o king, live forever, let not thy thoughts trouble thee, nor let thy countenance be changed; there is a man in thy kingdom, in whom is the spirit of the holy gods; and in the days of thy father light and understanding and wisdom,like the wisdom of the gods, was found in him; whom the king Nebuchadnezzar thy father, the king, Isay thy father, made master of the magicians, astrologers, chaldeans, and soothsayers;forasmuch as an excellent spirit, and knowleage, and understanding, interpreting of dreams, and showing of hard sentences, and dissolving of doubts, were found in the same Daniel, whom the king named Belteshazzar; now let Daniel be called, and he will show the interpretation. The first codeword is ohtello.

加密函数为

$$f = \begin{pmatrix} \text{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\ \text{MFANXIWPBSHGLTCQKVRJUDYZEO} \end{pmatrix}$$

2.4 福尔摩斯密码

破译密码，本来就是抽丝剥茧的过程，这与侦探小说的内容有异曲同工之妙，在闻名的侦探小说家柯南道尔爵士（Sir Conan Doyle）笔下的名侦探福尔摩斯（Sherlock Holmes）的著名案件里，其中的一篇《跳舞小人历险记》（the Adventure of the Dancing Men），便是利用破解密码（单套字母替代法）来找到杀人凶手而破案的。故事的内容简述如下：在英国 Riding Thorpe Manor 的 Hilton Cubitt 先生最近娶了来自美国芝加哥的 Elsie Patrick。在家里的花园中，Cubitt 发现了一张如图 2-1 所示画有跳舞小人的字条，于是他就将这张字条寄给他的朋友福尔摩斯。两周后，Cubitt 又在他家的工具室里发现有人用粉笔在门上画了类似的符号，如图 2-2 所示，他就将这些符号抄录下来寄给福尔摩斯。三天后，另一个跳舞小人的信息又在家里出现，如图 2-3 所示，接着又出现如图 2-4、图 2-5 所示的跳舞小人。Cubitt 百思不解，他总觉得有人要传递什么信息给他或者他的夫人。于是他把所有的跳舞小人全部寄给了福尔摩斯，希望福尔摩斯能帮他找到答案。



图 2-1　跳舞的小人 1

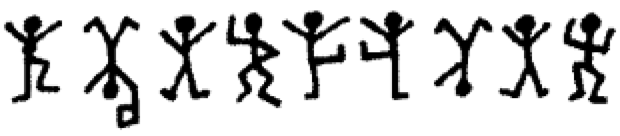


图 2-2　跳舞的小人 2



图 2-3　跳舞的小人 3



图 2-4　跳舞的小人 4

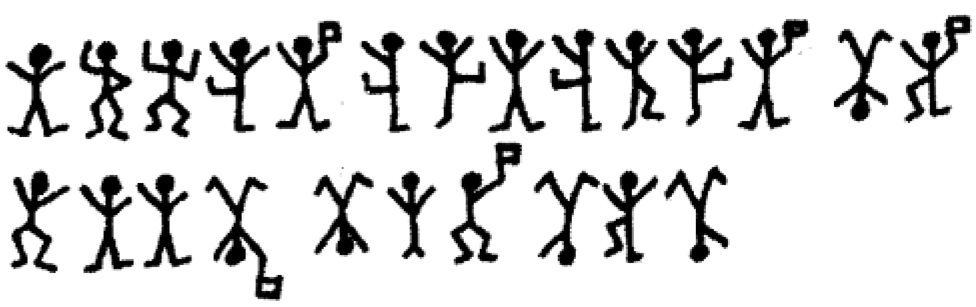


图 2-5　跳舞的小人 5

福尔摩斯研究后认为这些跳舞的小人代表着有某种有意义的信息。福尔摩斯仔细研究这些不同姿势的跳舞小人出现的频率后，发现大事不妙，于是他立即前往 Cubitt 家中。没想到在他抵达时，警察已经在那里了，因为 Cubitt 已被枪杀身亡，而他的太太 Elsie 也受了重伤并且紧紧地躺在他身边。在询问了她几个问题后，福尔摩斯终于了解了整个案情的来龙去脉，于是福尔摩斯写了一张纸条，派人送给 Abe Slaney 先生。在警察问到为何有如此的动作时，福尔摩斯开始解释他是如何破解这些跳舞小人信息的。

他说在 Cubitt 寄给他的信中，就如图 2-6 所示的符号出现的次数最多，所以他便假设这个符号代表英文字母中的 E。如此一来，在这些信息中的其中一对跳舞的小人如图 2-7 所示便是 -E-E-，而可能对应的单词是 LEVER（杠杆）、NEVER（决不）、SEVER（分开）。于是福尔摩斯凭直觉猜测这个词是 NEVER。所以如图 2-8 所示的跳舞的小人便代表着 N、V、R。



图 2-6　跳舞的小人 6



图 2-7　跳舞的小人 7



图 2-8　跳舞的小人 8



图 2-9　跳舞的小人 9

当警察问到是否要去抓 Abe Slaney 时，福尔摩斯笑道：“不用！他一会儿就会自己来了。”原来，Abe Slaney 是 Elsie 在芝加哥的未婚夫，他深爱着 Elsie，然而 Elsie 已经厌倦了跟着父亲的帮派生活，在知道 Abe Slaney 是怎么样的一个人后，Elsie 逃到英国，并在后来嫁给了心爱的 Hilton Cubit。Abe Slaney 发现了 Elsie 的踪迹后，便心生妒嫉，开枪杀害了 Cubitt。但 Abe Slaney 万万没想到的是，他深爱的 Elsie，居然愿意与 Cubitt 共生死。而这些“跳舞的小人”便是这个帮派中用来传递信息的符号。当大家再度很好奇地问福尔摩斯的信上写着什么时，福尔摩斯又以他闻名的笑容，在纸上写着：“COME HERE AT ONCE”。然后便消失在人群当中。

2.5　Vigenère 密码*

16 世纪，法国的 Vigenère 发明了一种多套字母替代（Polyalphabetic Substitution）算法，即 Vigenère 密码，它与单套字母替代法最大的不同就是加密时不是只针对单一字母加密，而是一个区域一个区域加密，其加密方法可整理如下。

算法 2.1　（Vigenère 密码）：令区块的长度为 d，其中信息代码为

$$x=(x_1,x_2,\cdots,x_d)\in(\mathbb{Z}/26)^d$$

密钥为

$$k=(k_1,k_2,\cdots,k_d)\in(\mathbb{Z}/26)^d$$

则加密函数为

$$E(x)=x+k=(x_1+k_1,x_2+k_2,\cdots,x_d+k_d)\pmod{26}$$

而解密函数为

$$D(y)=y_k=(y_1-k_1,y_2-k_2,\cdots,y_d-k_d)\pmod{26}$$

当中加密函数与解密函数互为反函数，即

$$\begin{cases}E(D(y))=y\\D(E(x))=x\end{cases}$$

事实上，Vigenère 密码是一种挪移密码，与凯撒挪移密码不同之处，只是挪移的是字母区块而非单一字母。

例 2.8　Alice 欲使用 Vigenère 密码加密法与 Bob 通信，其中他们已事先约定双方用密钥

$$k=(21,4,2,19,14)。$$

而明文为 m='ciphertext'。其加密过程为：

明文	c	i	p	h	e	r	t	e	x	t
明文代码	2	8	15	7	4	17	19	4	23	19
密钥	21	4	2	19	14	21	4	2	19	14
模加法值	23	12	17	0	18	12	23	6	16	7
密文	X	M	R	A	S	M	X	G	Q	H

Vigenère 密码在早年，是被视为牢不可破的，其主要的原因，是未找出如何“猜”出密码的区域长度的方法；一旦区域长度能确定，就可运用针对单套替代法的频率分析，找出密钥值，

因此密文攻击方面主要要考虑：

- 找出区域的长度，即密钥的长度。
- 找出个别的密钥值。

一种找出密钥长度的方式，就是将密文印成两份工整的长条，固定其中一份，移动另一份密文长条比对上下密文字母相吻合之处，并将其吻合字母数写下，再移动密文长条的位置，比对、记下吻合字母数，如此，比对不同的位置为止，而其中最高吻合对数的移动位置，应是密钥的长度。这种找出密钥长度的方法，其实很容易程序化，可参考下列算法。

算法 2.2 计算可能密钥长度：

```
int coincidence ( char * ciphertext , int place )
{
//计算不同位置的吻合字母数
    int count = 0 ;
    int j ;
    int n = strlen ( ciphertext ) ;
    for ( inti = 0 ; i < n ; i + + )
        {
            j = i + place ;
            if  ( j > = n )
                break ;
            if ( ciphertext [ i ] = = ciphertext [ j ] )
                count + + ;
        };
    return count;
};
int keyLength ( char * ciphertext )
{ //计算可能密钥长度
    int keyLength = 0 ;
    int j ;
    for ( int  i = 1 ; i < = 13 ; i + + )
    {
        j = coincidence ( ciphertext , i ) ;
        if ( j > keyLength )
            keyLength = j ;
    } ;
    return keyLength ;
};
```

所要注意的，使用这种方式寻找密钥的长度，要先假设密钥的长度≤13。这种找密钥长度的方式，主要是依据不同字母出现频率的比率，考虑向量空间 $\mathbb{R}^{26}$ 的向量 W（为字母出现频率的比率）

$$
\begin{aligned}
W &= (p(a),p(b),p(c),\cdots p(z),) \\
&\approx (0.082,0.015,0.028,0.043,0.127,0.022,0.020, \\
&\quad 0.061,0.070,0.002,0.008,0.040,0.024,0.067, \\
&\quad 0.075,0.019,0.001,0.060,0.063,0.091,0.028, \\
&\quad 0.010,0.023,0.001,0.020,0.001)
\end{aligned}
$$

假设 V 也是 $\mathbb{R}^{26}$ 中的向量，且 V 的选择为向量 W 的各坐标值的置换。由于

$$\cos\theta = \frac{W \cdot V}{|W||V|}$$

其中 θ 为向量 $W=(w_1,\cdots,w_{26})$ 与 $V=(v_1,\cdots,v_{26})$ 的夹角弧度，$|\bullet|$ 表示向量的长度，如

$$|W| = \sqrt{w_1^2 + w_2^2 + \cdots + w_{26}^2}\ 。$$

且各个字母出现的频率均不同，可得

$$\begin{aligned} W \bullet V &\leqslant W \bullet W \\ &\approx (0.082)^2 + (0.015)^2 + \cdots + (0.001)^2 \approx 0.066 \end{aligned}$$

其中，只有在 $W \approx V$ 之时 ≈ 才会成立。取 W_i 为向量 W 右旋 i 个位置，可得内积值如下表：

$\|i-j\|$	0	1	2	3	4	5	6
$W_i \cdot W_j$	0.066	0.039	0.032	0.034	0.044	0.033	0.036
$\|i-j\|$	7	8	9	10	11	12	13
$W_i \cdot W_j$	0.039	0.034	0.034	0.038	0.045	0.039	0.042

而 $|i-j|$ 反映了在移动密文长条的位置时，所得的吻合对应数为

$$\approx n \times W_i \cdot W_j\,,$$

其中 n 为密文的长度，最高的吻合对数约为 n×0.066，这是其他吻合对数的大约两倍。在 1920 年，Wolfe Friedman 就曾定义所谓吻合指数（Index of Coincidence）：

定义 2.2　令 $x = x_1x_2\cdots x_n$ 为长度为 n 的字母字串，定义此字串 x 的吻合指数（Index of Coincidence）为任取此字串两字母为相同的几率，即

$$I(x) = \frac{\sum_{i=1}^{26}\binom{p_i}{2}}{\binom{n}{2}}$$

其中 $p_1, p_2, \cdots, p_{26}$ 代表字母 $a, b, c, \cdots z$ 在字串 x 中所出现的次数。

假设 x 为一篇普通的英文文章，且 n 很大，则

$$I(x) \approx |W| = \sqrt{w_1^2 + w_2^2 + \cdots + w_{26}^2} \approx 0.066$$

若 x 为经 Vigenère 方法加密后的密文，可将字串 x 分割成 m 个 $\lceil n/m \rceil$ 的子字串，即 $y_1, y_2, \cdots, y_m$，其中

$$\begin{aligned} y_1 &= x_1 x_{m+1} x_{2m+1} \cdots, \\ y_2 &= x_2 x_{m+2} x_{2m+2} \cdots, \\ y_3 &= x_3 x_{m+3} x_{2m+3} \cdots, \\ &\ddots \\ y_m &= x_m x_{2m} x_{3m} \cdots, \end{aligned}$$

若 m 恰好为密钥的长度（或其整数倍），则

$$I(y_i)=0.066,(i=1,2,\cdots,m),$$

否则

$$I(y_i)=26\left(\frac{1}{26}\right)^2=\frac{1}{26}\approx 0.038,$$

这就是为何密钥长度算法能够进行的原因。聪明的读者也可借的吻合指数这个概念，推导出更为精细的频率分析法，用以估算密钥的长度。

例 2.9 有 Vigenère 密文不知其密钥长度注[3]，欲破译此密文。

```
ciphertext=
"O C W Y I K O O O N I W U G P M X W K T Z D W G
T S S A Y J Z W Y E M D L B N Q A A A V S U W D
V B R F L A U P L O O U B F G Q H G C S C M G Z
L A T O E D C S D E I D P B H T M U O V P I E K
I F P I M F N O A M V L P Q F X E J S M X M P G
K C C A Y K W F Z P Y U A V T E L W H R H M W K
B B V G T G U V T E F J L O D F E F K V P X S G
R S O R V G T A J B S A U H Z R Z A L K W U O W
H G E D E F N S W M R C I W C P A A A V O G P D
N F P K T D B A L S I S U R L N P S J Y E A T C
U C E E S O H H D A R K H W O T I K B R O Q R D
F M Z G H G U C E B V G W C D Q X G P B G Q W L
P B D A Y L O O Q D M U H B D Q G M Y W E U I K"
```

- （密钥长度）代入程序 coincidence（ciphertext，place），如表 2-4 所示，故猜密钥长度为 6。

表 2-4 计算吻合数 coincidence (ciphertext , place)

place	1	2	3	4	5	6	7	8	9	10	11	12	13
coincidence	13	13	11	6	15	23	8	5	12	11	10	18	10

- （破译密钥）由于密文很短，出现字母最多的，不一定其明文是"e"、"t"或"a"，与其瞎猜不如用更为精细的频率分析。考虑密钥的第一坐标值：令 C[1]为 ciphertext 的子字串，由第 1，7，13，…，6k+1，…个字母生成，字母出现频率如表 2-5 所示。

注[3] 该密文摘自 Wade Trappe 和 Lawrence C.Washington 的"Introduction to Cryptography With Coding Theory"57 页的习题。难度比《码书》的第四个密文 Vigenere 密文大多了。

表 2-5　ciphertext 的 C[1]子字串的字母出现频率分析表

字母	出现次数	出现频率比例	字母	出现次数	出现频率比例
A	3	0.058	N	3	0.058
B	4	0.077	O	4	0.077
C	2	0.038	P	4	0.077
D	0	0	Q	0	0
E	0	0	R	1	0.019
F	1	0.019	S	1	0.019
G	0	0	T	2	0.038
H	5	0.096	U	7	0.134
I	2	0.038	V	1	0.019
J	1	0.019	W	2	0.038
K	3	0.058	X	0	0
L	4	0.077	Y	1	0.019
M	0	0	Z	1	0.019

令

$$\begin{aligned}P &= (p(A), p(B), \cdots, p(Z)) \\ &\approx (0.058, 0.077, 0.038, 0, 0, 0.019, 0, 0.096, 0.038, 0.019, \\ &\quad 0.058, 0.077, 0, 0.058, 0.077, 0.077, 0, 0.019, 0.019, 0.038, \\ &\quad 0.134, 0.019, 0.038, 0, 0.019, 0.019)\end{aligned}$$

为 C[1]各字母所出现的比例，向量 W 表示一般英文字母出现频率比例，而 W_{-i} 为将向量 W 左旋 i 位置，计算内积值

$$P \cdot W_{-i} \quad (i = 0, 1, 2, \cdots, 25)$$

表 2-6　$P \cdot W_{-i}$ 内积值

i	0	1	2	3	4	5	6
$P \cdot W_{-i}$	0.042	0.045	0.037	0.045	0.029	0.030	0.042
i	7	8	9	10	11	12	13
$P \cdot W_{-i}$	0.056	0.038	0.040	0.035	0.037	0.032	0.039
i	14	15	16	17	18	19	20
$P \cdot W_{-i}$	0.041	0.039	0.041	0.032	0.035	0.038	0.048
i	21	22	23	24	25		
$P \cdot W_{-i}$	0.032	0.049	0.039	0.033	0.030		

发现在 i = 7 之时内积值 0.056 最大，因此向量 P 与 W_{-7} 最接近，故猜密钥的第一向量为 7。如此方法继续进行，可得完整的密钥值

$$(7, 14, 11, 12, 4, 10)$$

同时考虑解码、标点和空格，可得明文。

- （明文）明文原来为“The Adventure of the Dancing Men”的第一段文字：
  Holmes had been seated for some hours in silence with his long, thin back curve dover a chemical vessel in which he was brewing a particularly malodorous product. His head was sunk upon his breast, and he looked from my point of view like a strange, lank bird, with dull grey plumage and ab lack top-knot. “So Watson,” said he, suddenly,” you do not propose to invest in South African securities?”

2.6 Hill 密码

在 1929 年，《美国数学月刊（American Mathematical Monthly）》杂志上出现了一篇标题为“Cryptography in an Algebraic Alphabet”的文章，作者为 Lester Hill，在这篇文章中，Hill 引入了一种利用矩阵运算多套字母替换的加密方式，这种加密方式在当今大部分的线性代数课本中也当作教材，其主要概念就是利用系数为 $\mathbb{Z}/26$ 的矩阵乘法，其作法可见下例：

例 2.10 Alice 要利用 Hill 密码将明文 m=‘Hill’加密成密文 c，传递给 Bob。

- （密钥产生）首先决定所用矩阵的大小，譬如是 2×2，

$$E=\begin{bmatrix}1 & 1\\ 3 & 4\end{bmatrix},$$

其中 E 的行列式值 detE 必须与 26 互质，否则不存在 E 的反矩阵。

- （加密）Alice 将明文 m=‘Hill’写成矩阵形态，即

$$M=\begin{bmatrix}h & l\\ i & l\end{bmatrix}=\begin{bmatrix}7 & 11\\ 8 & 11\end{bmatrix},$$

（因已经约定明文代码 $a=0, b=1, \cdots, z=25$）将加密矩阵 E 左乘明文代码矩阵 M，得

$$\begin{aligned}EM&=\begin{bmatrix}1 & 1\\ 3 & 4\end{bmatrix}\begin{bmatrix}7 & 11\\ 8 & 11\end{bmatrix}\\ &=\begin{bmatrix}15 & 22\\ 53 & 77\end{bmatrix}\\ &=\begin{bmatrix}15 & 22\\ 1 & 25\end{bmatrix} \pmod{26}\end{aligned}$$

故得密文代码矩阵

$$C=\begin{bmatrix}15 & 22\\ 1 & 25\end{bmatrix}=\begin{bmatrix}p & w\\ b & z\end{bmatrix},$$

将密文 c=‘pbwz’传给 Bob。

- （解密）Bob 计算加密矩阵的反矩阵，再进行模运算（mod 26），得解密矩阵

$$D=\begin{bmatrix}4 & -1\\ -3 & 1\end{bmatrix} \pmod{26},$$

将所得的密文代码矩阵 C，以解密矩阵 D 左乘，得

$$EM = \begin{bmatrix} 4 & -1 \\ -3 & 1 \end{bmatrix}\begin{bmatrix} 15 & 22 \\ 1 & 25 \end{bmatrix}$$
$$= \begin{bmatrix} 59 & 53 \\ -44 & -41 \end{bmatrix}$$
$$= \begin{bmatrix} 7 & 11 \\ 8 & 11 \end{bmatrix} \pmod{26}$$
$$= \begin{bmatrix} h & l \\ i & l \end{bmatrix}$$

得明文 m=‘Hill’。

Hill 密码是不敌明文攻击的。

例 2.11（续上例）　假设攻击者 Eve 已取得一组对照文 $(m,c) = ('hill','pbwz')$，欲破译其解密函数 $D(c) = m$，就是解矩阵同余方程

$$D = \begin{bmatrix} d_{11} & d_{12} \\ d_{13} & d_{14} \end{bmatrix}$$
$$\equiv MC^{-1} = \begin{bmatrix} 7 & 11 \\ 8 & 11 \end{bmatrix}\begin{bmatrix} 15 & 22 \\ 1 & 25 \end{bmatrix}^{-1} \pmod{26}$$
$$= \begin{bmatrix} 7 & 11 \\ 8 & 11 \end{bmatrix}\left(\frac{1}{353}\begin{bmatrix} 25 & -22 \\ -1 & 15 \end{bmatrix}\right)$$
$$\equiv 15^{-1}\left(\begin{bmatrix} 7 & 11 \\ 8 & 11 \end{bmatrix}\begin{bmatrix} -1 & 4 \\ -1 & 15 \end{bmatrix}\right) \pmod{26}$$
$$\equiv 7\begin{bmatrix} -18 & 193 \\ -19 & 197 \end{bmatrix} \pmod{26}$$
$$= \begin{bmatrix} 4 & 25 \\ 23 & 1 \end{bmatrix} \pmod{26}\text{。}$$

至于密文攻击，是较为复杂的。第一步就是要先决定所用加密解密矩阵的大小，这同样可以利用找出 Vigenère 密码密钥长度的方法；而决定加密解密的方法就是利用破译单套字母替换的方法，找出每区块特定位置的代换关系，并解出代数式；这些都是频率分析结合矩阵运算所能破解的，故 Hill 密码并无多大实用价值，不过是人类在发展密码技术过程中运用代数计算的一段小插曲。

2.7　单次密码本

最著名的无法破解的密码系统，应属单次密码本（one-Time Pad ），早在 1918 年前后，就由 Gilbert Vernam 以及 Joseph Mauborgne 给出，这种密码系统在冷战期间为苏联间谍所使用。单次密码本的密码不可重复使用，否则就有破译的可能，其加密解密方式可见以下例子。

例 2.12　Aice 欲将明文 m=‘OK’，借由单次密码本加密成密文 c，传递给 Bob。假设密钥为 k=‘DA’。

- （加密）Aice 先将明文 m 化为 ASCII 代码（当然双方必先协定所用的代码方式）并加密，即

$$
\begin{array}{lr}
\text{明文}\,m = \text{'OK'} = (79,75)_{ASCII} = 10011111001011 & \\
\text{密钥}\,k = \text{'DA'} = (68,65)_{ASCII} = 10001001000001 & \oplus \\
\hline
\text{密文}\,c = \qquad\qquad\qquad\qquad 00010110001010 &
\end{array}
$$

将密文 c=00010110001010 传给 Bob。

- （解密）Bob 取得密文，并用密钥解密：

$$
\begin{array}{lr}
\text{密文}\,c = \quad 00010110001010 & \\
\text{密钥}\,k = \ 10001001000001 = \text{'DA'} & \oplus \\
\hline
\text{明文}\,m = \ 10011111001011 = \text{'OK'} &
\end{array}
$$

（其中⊕表示 XOR 运算，即

$$1 \oplus 0 = 0 \oplus 1 = 1, 0 \oplus 0 = 1 \oplus 1 = 0$$

注意：英文 26 个大写字母只占 ASCII 码很小的一部分，因此只用小部分 ASCII 码的编码方式并不恰当。另外，密钥必须由适当的随机数生成器产生（至少是伪随机数生成器），否则也有被“猜”中密钥的危险。

单次密码本的 XOR 运算也可修改成一般模加法运算，如下例所示。

例 2.13 Alice 欲将明文 m=‘OK’，借由（mod 26）版单次密码本加密成密文 c，传递给 Bob。并假设密钥为 k=‘DA’。

- （加密）Alice 将明文加密：

$$
\begin{array}{lr}
\text{明文}\,m = \text{'OK'} = (14,10) & \\
\text{密钥}\,k = \text{'DA'} = (3,0) & + \\
\hline
\text{密文}\,c = \ \text{'RK'} \ = (17,10) \ (\text{mod } 26) &
\end{array}
$$

- （解密）Bob 将密文解密：

$$
\begin{array}{lr}
\text{密文}\,c = \ \text{'RK'} \ = (17,10) & \\
\text{密钥}\,k = \text{'DA'} = (3,0) & + \\
\hline
\text{明文}\,m = \text{'OK'} = (14,10) \ (\text{mod } 26) &
\end{array}
$$

2.8 Enigma 密码机

二次世界大战期间，同盟国适时地破译德国 Enigma 密码机所编出的密文，几乎解读了德国当时所有重大的政治、军事、外交密电；即使二次世界大战末期德国发展出类似 Enigma 却比 Enigma 更强的 Lorenz 密码机来传递希特勒与高级领导之间的密电，同盟国也能利用 Colossus 破译机破译，如图 2-10 所示。在几乎毫无国防机密可言的情况下，纳粹第三帝国输了战役也亡了政权，矢志要将犹太人灭种的希特勒，也在他的领袖防空洞内举枪自尽。让我们将场景转到二次世界大战初期，Enigma 密码机伴随着德国装甲师团以及 U 艇出征，通过它与

远在柏林的最高国防军统帅部（Oberkommando der Wellmlacht）联系，加密、解密信息，然而这些摩斯码信息极易遭截收，远在英国 Bletchiey Park 的密码研究员尝试分析这些截收到的密文找出密文的特征模式，再利用他们所发明制造的破译机 Bombes 计算，不需几小时，只要所截收的密文数够多，就可以破解当日 Enigma 密码机所用的密钥。这就是 Alice-Eve-Bob 场景的二次大战版。

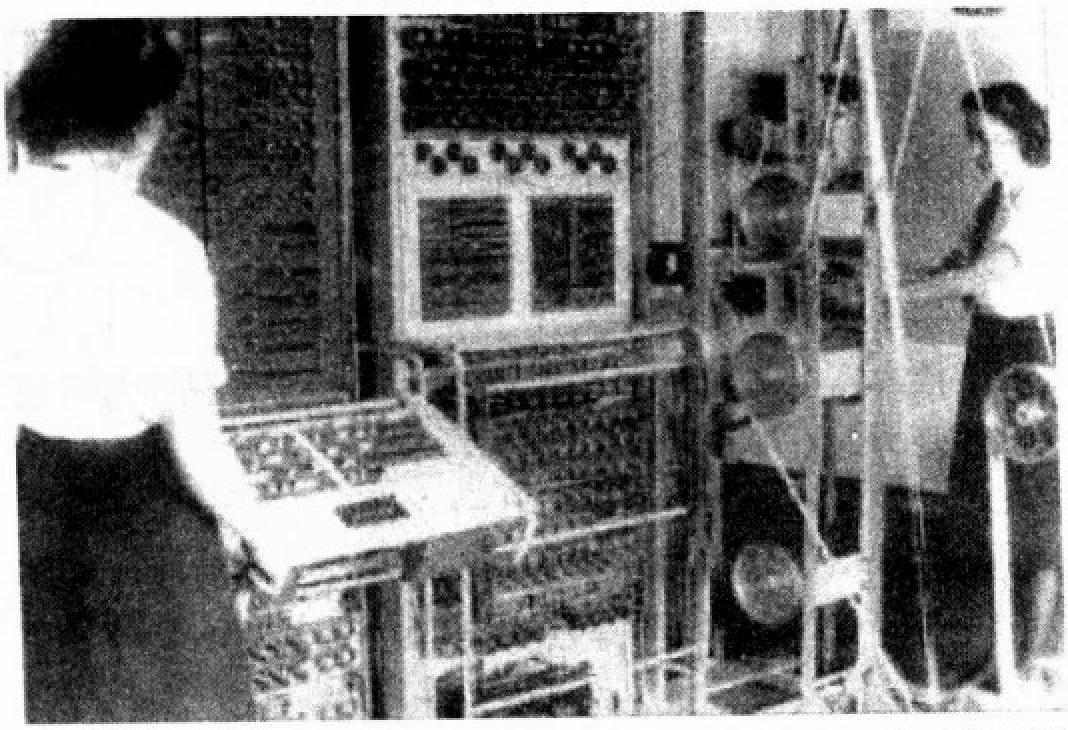


图 2-10　Max Newman 设计的人类首部电脑——Colossus 破译机，战后连同蓝图一起销毁（取自 www.bletchleypark.org.uk）

1918 年，德国发明家 Arthur Scherbius 发明的 Enigma 密码机由多项零件所组合而成，主要结构为：

- 键盘：为明文字母输入用。
- 接线板（Stecker）：为字母置换，早期有 6 条接线，可作 6 对字母间的置换。
- 滚轮（Roter）：早期有 3 个，可改变位置，为 Enigma 密码机最关键的部位，相当于 3 个编码器。
- 反射器（Reflector）：固定式的置换作用，使得 Enigma 密码机加密与解密的程序一致，故 Enigma 密码为对称密钥密码。
- 灯板：为密文字母输出用。
- 电源器：可提供直流/交流电源输入。

可参阅 Enigma M3 型密码机编码路径如图 2-11 所示，其中 L、M、N 分别代表 3 个不同的滚轮，R 表示反射器，S 为接线板，K 代表键盘，B 代表灯板。每个滚轮有类似 26 齿的齿轮的结构以及牵动其他轮的凹槽（Notch），只要一按键盘任意键，滚轮 N 就会顺时针方向转动 1/26 圈一格，N 转一周（或至凹槽），滚轮 M 才会转动一格，即 1/26 圈，当滚轮转一周（或至凹槽），滚轮 L 才会转动一格，即 1/26 圈，这种滚轮互相牵动的情形类似水表的变动[注4]。

[注4] 实际上滚轮互相牵动的实际情况并不是与水表“进位”转动完全一样，由于特殊的机械关系，连续按键两次，滚轮 M 有时会被牵动两格，直到 G 型才加入齿轮箱，解除这种现象。

而编码是在滚轮牵动之后，Enigma 密码机的滚轮装置使得明文字母与密文字母的关联几乎无迹可寻。而 Enigma 密码机的“密钥”其实是指（以战争初期 M3 版而言）：

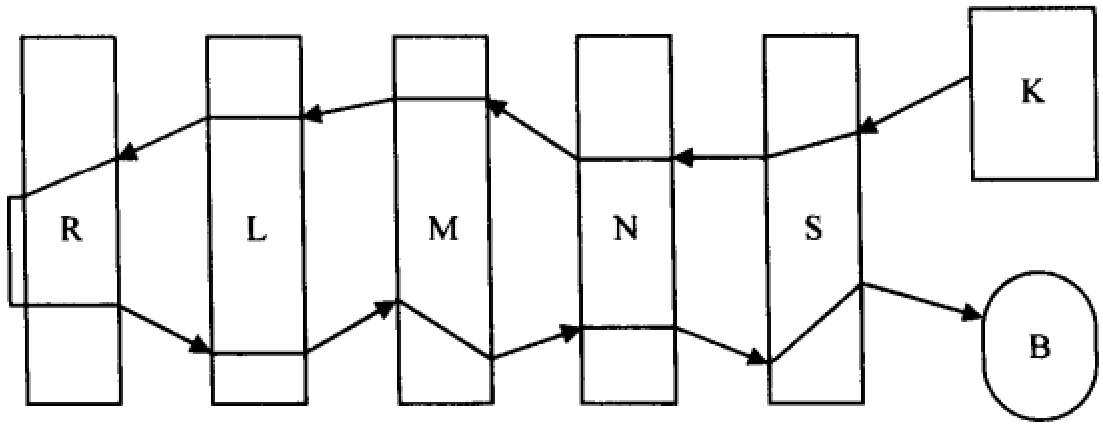


图 2-11 Enigma M3 型密码编码路径图

- 接线板上 6 条接线连接几对字母（相连的两字母即表示两字母对换），以接线板上的 A 与 B 接线相连为例，即

$$A \to B \text{ 与 } B \to A,$$

为方便启用，以 2-Cycle（AB）表示。使用 0、1、…、6 条接线，共有

$$\begin{aligned}
&\binom{26}{2}\binom{24}{2}\binom{22}{2}\binom{20}{2}\binom{18}{2}\binom{16}{2}\Big/6!\\
&+\binom{26}{2}\binom{24}{2}\binom{22}{2}\binom{20}{2}\binom{18}{2}\Big/5!\\
&+\binom{26}{2}\binom{24}{2}\binom{22}{2}\binom{20}{2}\Big/4!\\
&+\binom{26}{2}\binom{24}{2}\binom{22}{2}\Big/3!\\
&+\binom{26}{2}\binom{24}{2}\Big/2!\\
&+\binom{26}{2}\\
&+1\\
&=100391791500+5019589575+164038875\\
&\quad+3453450+44850+325+1\\
&=105578918576
\end{aligned}$$

种可能性。

- 3 个滚轮的位置，以及滚轮上初始字母，共有

$$3!(26)^3=105456$$

种可能性。

- 故有

$$105456\times105578918576=11133930437350656\approx10^{16}$$

种可能性。这在以每分钟可以检查一种密钥的速度下需要比宇宙岁数还要长的时间才能检查完毕。[注[5]]

以下为简化版的 Enigma 密码机编码说明（假设无“进位”现象，只有最右轮转动）：

例 2.14　假设滚轮的顺序为 1、2、3，而滚轮的初始位置为 AAA，接线板上不用任何接线，故接线板上的置换作用为 s=id，输入明文为 AA，则密文为 FH。试解释其加密路径如下：

（1）滚轮与反射器的置换为

$$\text{Rotor1}: l=\begin{pmatrix}\text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}\\ \text{EKMFLGDQVZNTOWYHXUSPAIBRCJ}\end{pmatrix}$$

$$\text{Rotor2}: m=\begin{pmatrix}\text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}\\ \text{AJDKSIRUXBLHWTMCQGZNPYFVOE}\end{pmatrix}$$

$$\text{Rotor3}: n=\begin{pmatrix}\text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}\\ \text{BDFHJLCPRTXVZNYEIWGAKMUSQO}\end{pmatrix}$$

$$\text{Reflector}: r=\begin{pmatrix}\text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}\\ \text{YRUHQSLDPXNGOKMIEBFZCWVJAT}\end{pmatrix}$$

（2）按下明文字母第一个 A，由于接线板无接线，A 保持不变，按键使得 Rotor3 向前滚动一格（系 Rotor3 的第一列向右旋一格），此时

$$\text{Rotor3}: n_1=\begin{pmatrix}\text{ZABCDEFGHIJKLMNOPQRSTUVWXY}\\ \text{BDFHJLCPRTXVZNYEIWGAKMUSQO}\end{pmatrix}$$

此时的置换作用为

$$s\circ n_1^{-1}\circ m^{-1}\circ l^{-1}\circ r\circ l\circ m\circ n_1\circ s$$

因此

$$A\to D,$$

然而 Rotor2、Rotor1 皆不滚动，因此在 Rotor2 上，

$$D\to K,$$

在 Rotor1 上

$$K\to N,$$

在反射器上

$$N\to K,$$

回到 Rotor 上

$$K\to B,$$

（原先的反作用，第二列值对回第一列值），回到 Rotor2 上

$$B\to J,$$

回到 Rotor3 上

$$J\to F,$$

最后经由接线板 $F\to F$。

注[5]　其实 Enigma 还有一项装置，称为环（Ring），这也会增加密密钥的数目。这项装置可调整滚轮的“进位”位置，是 Enigma 装置中最不重要的。

（3）按下明文字母第二个 A，按键使得 Rotor3 再向前滚动一格（原 Rotor3 的第二列向右旋二格），此时

$$\text{Rotor3}: n_2 = \begin{pmatrix} \text{YZABCDEFGHIJKLMNOPQRSTUVWX} \\ \text{BDFHJLCPRTXVZNYEIWGAKMUSQO} \end{pmatrix}$$

此时的置换作用为

$$s \circ n_2^{-1} \circ m^{-1} \circ l^{-1} \circ r \circ l \circ m \circ n_2 \circ s \text{。}$$

经由接线板因无接线，A 保持不变，在 Rotor3 上

$$A \to F\text{，}$$

Rotor2 上

$$F \to I\text{，}$$

Rotor1 上

$$I \to V\text{，}$$

到反射器上

$$V \to W\text{，}$$

回到 Rotor1 上

$$W \to N\text{，}$$

回到 Rotor2 上

$$N \to T\text{，}$$

回到 Rotor3 上

$$T \to H\text{。}$$

读者请注意，反射器的置换作用其实为

$$\begin{aligned} r &= (AY)(BR)(CU)(DH)(EQ)(FS)(GL)(IP)(JX)(KN)(MO)(TZ)(VW) \\ &= r^{-1}\text{，} \end{aligned}$$

它是 13 个两两不相交 2-Cycle（或称之为 Transposition）的乘积，与其反作用 r^{-1} 是相同的，加入反射器装置，使得加密解密程序相同；另外也由于反射器的装置，明文字母一定被编成不同的密文字母[注[6]]。有关 Enigma 更进一步的资料可参阅以下网址：

- Bletchley Park 官方网站：
  http://www. Bletchleypark.org.uk，该网站除了说明 Enigma 密码机的历史沿革，更提供了一个 Enigma 密码机模拟器，读者可自行上线操作。
- 有关 Alan Turing 的网站：
  www.turing.org.uk/turing
  home.cern.ch/f/frode/www/crypto/Turing/index.html
  当中有 Turing 当年是如何破译 Enigma 的详尽描述。
- 对于波兰对破译 Enigma 的贡献，不能只从好莱坞影片上了解，可参阅：
  home.us.net/~encore/Enigma
- 与 Bletchley Park 相关的网址为：

注[6] 这些都可视为群论中对称群的习题，参见下节“破译 Enigma 与对称群”的定理 2.1。

www.codesandciphers.org.uk/

www.gchq.gov.uk/

Enlgma 密码机如图 2-12 所示。



图 2-12　Enigma 密码机（取自 www. Bletchleypark.org.uk）

另外，读者若是打开 Google 搜索引擎，以关键字搜寻“Enigma simulator”，将会发现 3 万多项结果，除去无效链接，可找到为数不少的 Enigma 密码机的虚拟模拟器。

2.9　破译 Enigma 与对称群

早在德国未发动战争的 20 世纪 30 年代，波兰人 Marian Rejewski、Henryk Zygalski 以及 Jerzy Ròzycki 就已经能破译 Enigma 密电了，同时这项技术也在 1939 年纳粹德国发动侵略波兰战争之前数个月转移到英国，此后才有 Bletchiey Park 破译 Enigma 密电的传奇，如图 2-13 所示。早期德国人使用 Enigma 密码机，是用当日的密钥锁上该信息的密钥，只是接线板的设定不变，为了避免传递错误，该信息密钥的三个字母在密文开头重复出现两次。然而重复出现是使用密码的大忌，曾在德国歌廷根大学（GÖttingen Universitat）数学系学习的 Rejewski 就此利用对称群的特殊性质找出破译的方法注[7]。对称群中有一个很简单的性质，然而这就是破译 Enigma 的关键。

定理 2.1　令 $A=\{1,2,\cdots,n\}$，对称群 $S_n:=\{f:A\to A\mid f \text{ 为} 1-1\}$ 注[8]，令 f、$\sigma\in S_n$ 为置换，其中 σ 可写成长度为 k 的 Cycle，即

$$\sigma=(i_1 i_2\cdots i_m\cdots i_k);$$

注[7] 可参考 Marian Rejewski 的“An Application of the Theory of Permutations in Breaking the Enigma Cipher.” Applicationes Mathematicae.16,No.4,Warsaw 1980。

注[8] S_n 对合成运算“0”有封闭性，且满足群的各项公理。

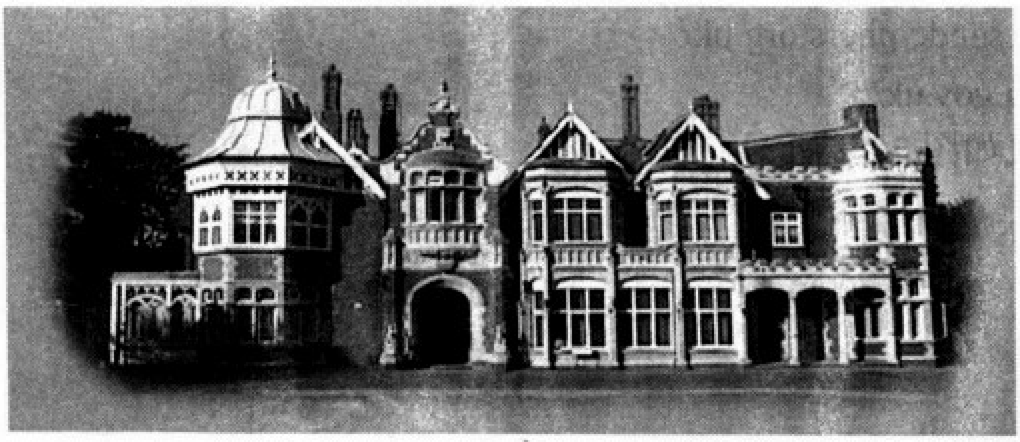


图 2-13　Bletchley Park（取自 www.bletchleypark.org.uk）

即

$$i_1 \mapsto i_2,\quad i_2 \mapsto i_3 \cdots i_{k-1} \mapsto i_k,\quad i_k \mapsto i_1,$$

其他不变。则

$$f \circ \sigma \circ f^{-1} = (f(i_1)f(i_2)\cdots f(i_m)\cdots f(i_k))$$

也是长度为 k 的 Cycle。

证明：只需观察下列交换图（Commutative Diagram）：

$$\begin{array}{ccc} i_m & \xrightarrow{\sigma} & i_m+1 \\ f^{-1}\uparrow & & \downarrow f \\ f(i_m) & \longrightarrow & f(i_m+1) \end{array}$$

推论 2.1　令 f、$\sigma \in S_n$ 为置换，其中 σ 可写成两两不相交（Disjoint）的 Cycle 乘积，其长度分别为

$$(k_1, k_2, \cdots, k_t)。$$

则 $f \circ \sigma \circ f^{-1}$ 也是两两不相交的 Cycle 乘积，其长度分别为

$$(k_1, k_2, \cdots, k_t)。$$

德国早期在 Enigma 加密操作上，是在密文开头重复两次 3 个字母的密钥，（该则信息的密钥是指滚轮编码器的初始位置），故第 1 个与第 4 个字母、第 2 个与第 5 个字母及第 3 个和第 6 个字母的明文相同。Enigma 在每次按键编码用不同的置换函数，假设依次为

$$\sigma_1、\sigma_2、\sigma_3、\cdots \in S_{26},$$

由定理 2.1 与反射器装置可知："Enigma"在每次按键所产生的置换函数必为 13 个两两不相交的 2-Cycle 乘积。而且每个明文字母一定会被编译成其他不同的字母。"

将截收到的密文，经过适当排序：

第 1 则密文：	D	M	Q	V	B	N	…
第 2 则密文：	V	O	N	P	U	Y	…
第 3 则密文：	P	U	C	F	M	Q	…
	⋮			⋮			

利用这些密文进行密码分析。

- 首先从第 1 则密文的第 1 个字母 D，连到第 1 则密文的第 4 个字母 V，即

$$D \to V,$$

- 从第 2 则密文的第 1 个字母 V 出发，连到第 2 则密文的第 4 个字母 P，即

$$V \to P,$$

- 从第 3 则密文的第 1 个字母 P 出发，连到第 3 则密文的第 4 个字母 F，即

$$P \to F,$$

$$\vdots$$

如此对应

$$D \to V \quad V \to P \quad P \to F \quad F \to K \quad K \to X$$
$$X \to G \quad G \to Z \quad Z \to Y \quad Y \to O \quad O \to D$$

形成Cycle（DVPFKXGZYO）。如果截收到的密文是够多，可以得到完整的置换函数，假设

$$\sigma_4 \circ \sigma_1 = (DVPFKXGZYOD)(EIJMUNQLHT)(BC)(RW)(S)(A)\text{。}$$

而置换 $\sigma_4 \circ \sigma_1$ 为长度分别为 $(10,10,2,2,1,1)$ 的 Cycle 乘积。

用类似的方法可以分析第 2 和第 5 个字母、分析第 3 个和第 6 个字母；假设分析第 2 个和第 5 个字母可能会得出长度为(9,9,3,3,1,1)的 Cycle 乘积的置换；分析第三和第六个字母时，可能会得到长度为(13,13)的 Cycle 乘积的置换。至此，虽然 Rejewski 还不知道当日密钥是什么，但他却了解这把 Enigma 钥匙会产生三套有如以下 Enigma 钥匙特征的置换：

- 第 1 和第 4 字母会得出：长度为 $(10,10,2,2,1,1)$ 的 Cycle 乘积的置换 $\sigma_4 \circ \sigma_1$，
- 第 2 和第 5 字母会得出：长度为 $(9,9,3,3,1,1)$ 的 Cycle 乘积的置换 $\sigma_5 \circ \sigma_2$，
- 第 3 和第 6 字母会得出：长度为 $(13,13)$ 的 Cycle 乘积的置换 $\sigma_6 \circ \sigma_3$。

值得注意的是，这三套 Enigma 钥匙特征置换中相同长度的 Cycle 会成对出现。注[9]

定理 2.2（Rejewski）　令 S_{26} 代表 26 个字母置换作用所形成的对称群。

（1）令置换函数 σ、$\tau \in S_{26}$，且均可化为 13 个两两不相交的 2-Cycle 乘积

$\Rightarrow$ 两置换函数的合成置换 $\sigma \circ \tau$ 中，相同长度的 Cycle 会成对出现。

（2）若置换函数 f 可表示为两两不相交的 Cycle 的乘积，且相同长度的 Cycle 会成对出现

$\Rightarrow f = \sigma \circ \tau$

其中，σ、τ 均可化为两两不相交的 2-Cycle 乘积。

证明：我们改写 Rejewski 的原始证明，用以说明第一个叙述。假设

$$\sigma = (a_2 a_3)(a_4 a_5)\cdots(a_{2k-2} a_{2k-1})(a_{2k} a_1)\cdots$$
$$\tau = (a_1 a_2)(a_3 a_4)\cdots(a_{2k-3} a_{2k-2})(a_{2k-1} a_{2k})\cdots\text{，}$$

其中 σ 中的 $(a_2 a_3)$，…，$(a_{2k} a_1)$ 与 τ 中的 $(a_1 a_2)$，…，$(a_{2k-1} a_{2k})$ 均为不相同的 2-Cycle。观察

注[9] 有的书上所举的例子有待商榷，如畅销书 Singh 的“The Code Book”中用以解释 Enigma 钥匙特征置换的例子，是不符合这个性质的。

$$a_1 \xrightarrow{(a_2a_3)(a_1a_2)} a_3\ ,$$
$$a_3 \xrightarrow{(a_4a_5)(a_3a_4)} a_5\ ,$$
$$\vdots$$
$$a_{2k-1} \xrightarrow{(a_{2k}a_1)(a_{2k-1}a_{2k})} a_1\ ,$$

以及

$$a_{2k} \xrightarrow{(a_{2k-2}a_{2k-1})(a_{2k-1}a_{2k})} a_{2k-2}\ ,$$
$$a_{2k-2} \xrightarrow{(a_{2k-4}a_{2k-3})(a_{2k-2}a_{2k-3})} a_{2k-4}\ ,$$
$$\vdots$$
$$a_2 \xrightarrow{(a_{2k}a_1)(a_1a_2)} a_{2k}\ ,$$

所以$\sigma \circ \tau$中会出现

$$(a_1a_3 \cdots a_{2k-3}a_{2k-1}) \text{与} (a_{2k}a_{2k-2} \cdots a_4a_2)$$

这样相同长度的 Cycle，因此在置换$\sigma \circ \tau$中，相同长度的 Cycle 会成对出现。

当然，Enigma 的特征显然是来自于当日钥匙的设定：例如接线板的设定、编码器位置及顺序和编码器方向的综合结果。假设当日钥匙原本在接线板里设定，要求调换 S 和 G 的路线，如果我们稍微修改一下当日钥匙的细节，把连接 S 与 G 的电线拿掉，改为连接 T 与 K，以调换 T 与 K 的路线，结果，分析第 1 个和第 4 个字母会得出如下的置换：（令置换 f = (SG)(TK)）

$$\begin{aligned}&\sigma_4' \circ \sigma_1' = (f \circ \sigma_4 \circ f^{-1}) \circ (f \circ \sigma_1 \circ f^{-1})\\ &= f \circ (\sigma_4 \circ \sigma_1) \circ f^{-1}\\ &= (DVPFTXSZYOD)(EIJMUNQLHK)(BC)(RW)(A)(G)\ 。\end{aligned}$$

（根据定理 2.1）虽然有一些 Cycle 的字母变了，可是每一条 Cycle 的长度并没有变，如推论 2.1。Rejewski 辨认出了一个只受滚轮编码器设定影响的 Enigma 钥匙特征。Rejewski 每天都会先记下所有截获信息的前 6 个字母，亦即重复加密的信息钥匙，然后建立其关联表，用以追寻字母置换与其 Enigma 钥匙特征。现在，Rejewski 可以翻开他那本 Enigma 钥匙特征目录了。所有的$3! \times 26^3$=105456 种编码器设定，都一一依据其 Enigma 钥匙特征详列在目录中，再进一步分析，就可解开滚轮编码器的顺序以及初始位置设定。

虽然接线板提供了 Enigma 最大钥匙总数，然而只要知道滚轮编码器的顺序以及初始位置设定，只要用相同的滚轮编码器设定（无论是哪种接线板设定），都可将原先的 Enigma 密文再编码成单套字母替换法的密文，这样，只须再用频率分析法（有时还不需要）就可破解。

后来德国改变重复密钥输入的方式，要破译Enigma密码，以完全如同上述一样的破译手法就无能为力了，需要用其他的方式了；不过，波兰这项技术却是其他破译方法的基础。

第 3 章　基础数论

当代各类密码技术，往往借助各类代数结构，如对称性密码系统 AES 以及各种以椭圆曲线为基础的公开密钥密码系统。前者在 Galois 域 $\mathbb{F}_{2^8}$ 上运行，同时也包括在 $\mathbb{F}_{2^8}$ 上的矩阵代数运算；后者则是运用椭圆曲线为交换群的实例。这些代数背景知识，往往对想要学习当代密码技术的人造成相当大的障碍。鉴于此，本书将从简易的模运算、辗转相除法开始，探讨乘法群 $\mathbb{Z}/n^{\times}$；并提及中国余式定理、原根、二次剩余、Galois 域以及质数理论和连分数等非常基础的理论基础。掌握了这些对于学习当代密码技术很有帮助。至于密码学用到的其他各类数论知识，如椭圆曲线、二次筛法等，将在其他相关章节予以探讨。另外我们将在本章讨论以数论问题为架构的“安全”伪随机数生成器（Pseudo Random Generator），这在近代密码学的应用中有相当重要的意义。

3.1　模运算与辗转相除法

本节将介绍简易模运算（Modular Operations）以及辗转相除法（Euclidean Algorithm），这些概念是密码学计算的基础，不可小觑；并引入相关代数术语，以深化其概念。首先考虑日常生活中的问题。

例 3.1　假设今天是星期五，请问 10000 天后是星期几？要回答此问题，可用模 7 加法：

$$5+10000 \pmod 7 \equiv 2 \quad \text{（即 5+10000 除以 7 的余数）}$$

所以得解，10000 天后是星期二。

另一个模运算常用的例子，就是一般书籍所用的 ISBN 码（International Standard Book Number），其实就是一种校验码（Detecting Error Code）。

例 3.2　以 Lilie 所著的《全民英语能力分级检定测验》一书为例，其 ISBN 码为

$$957-28898-8-5,$$

是一个 10 位的数码，最后一位 5 为校验码（如图 3-1 所示），而 Churchhouse 所著的“codes and ciphers”的 ISBN 码为

ISBN 957-28898-8-5

9 789572 889886

图 3-1　ISBN 码与条码（Barcode）

$$0-521-81054-X$$

校验码为 X。其实 ISBN 码用到模 11 计算（mod 11）：

- 任何一本书的 ISBN 码为 10 位的数码

$$S = a_1a_2a_3\cdots a_{10},$$

其中 a_1、…、$a_9 \in \{0,1,2,\cdots,9\}$，而 $a_{10} \in \{0,1,2,\cdots,9,X\}$，X 代表 10。

- ISBN 码必须满足

$$\sum_{i=1}^{10} ia_i \equiv 0 \pmod{11},$$

否则就是错误。

- 以 S=052181054X 为例

$$\begin{aligned}\sum_{i=1}^{10} ia_i &= 2\times5+3\times2+4\times1+5\times8+6\times1+8\times5\\ &\quad +9\times4+10\times10\\ &\equiv(-1)+6+4+(-4)+6+(-4)+3+1 \pmod{11}\\ &\equiv 0 \pmod{11}。\end{aligned}$$

模运算主要是以同余类（Congruence Class）为元素的运算。

定义 3.1（同余，Congruence）：令 $n \in \mathbb{N}$。令 a、$b \in \mathbb{Z}$ 为两整数，称 a 同余 b 模 n，记为 $a \equiv b \pmod{n}$，当 n 整除 $b-a$。而所有与 a 同余的整数所组成的集合，即

$$[a] = \{b \in \mathbb{Z} \mid a \equiv b \pmod{n}\}$$

称为 a 的同余类。所有同余类所形成的集合，即[注1]

$$\mathbb{Z}/n = \{[x] \mid x \in \mathbb{Z}\}。$$

注意：同余类[a]本身为一等价类（Equivalence Class），即满足：

（1）（反身性，Reflexivity）$\forall x \in [a]: x \equiv x$。

（2）（对称性，Symmetry）若 $x \equiv y$ 则 $y \equiv x$。

（3）（传递性，Transitivity）若 $x \equiv y$、$y \equiv z$ 则 $x \equiv z$。

例 3.3 $5 \equiv 16 \equiv -6 \pmod{11}$，令 $n=7$，则

$$[2] = \{x \in \mathbb{Z} \mid x \equiv 2 \pmod{7}\} = [9] = [-5] = [10005]。$$

$$\mathbb{Z}/7 = \{[0],[1],[2],[3],[4],[5],[6]\}。$$

在集合 $\mathbb{Z}/n$ 上可定义加法、减法以及乘法，而有时也可定义除法，但要看除数是否有乘法反元素，这些计算即模运算，以同余类的术语定义如下：

- $[a]+[b] := [a+b] = \{x \mid x \equiv a+b \pmod{n}\}$
- $[a]-[b] := [a-b] = \{x \mid x \equiv a-b \pmod{n}\}$
- $[a]\cdot[b] := [a\cdot b] = \{x \mid x \equiv ab \pmod{n}\}$

而在 $\mathbb{Z}/n$ 的加法运算满足以下性质：

（1）封闭性：若同余类 x、$y \in \mathbb{Z}/n$ 则 $x+y \in \mathbb{Z}/n$。

注[1] 不用符号 Z_n 是避免与 P-进位数（P-adic Number）混淆。

（2）交换律：若同余类 x、$y \in \mathbb{Z}/n$ 则 $x+y=y+x$。

（3）结合律：若同余类 x、y、$z \in \mathbb{Z}/n$ 则 $(x+y)+z=x+(y+z)$。

（4）存在加法单位素：存在 $0=[0]$，使得 $x+0=x=0+x$。

（5）存在加法反元素：对任一 $x \in \mathbb{Z}/n$ 存在 $-x$ 使得 $x+(-x)=0=(-x)+x$。

所以 $(\mathbb{Z}/n,+)$ 满足所有交换群（Abelian Group）的公理（Axiom），即 $(\mathbb{Z}/n,+)$ 为交换群；有时，在每一个同余类中各取一个元素形成代表系统（Representative System）代表 $\mathbb{Z}/n$，如 $\mathbb{Z}/3$ 可以代表系统 $\{0,1,2\}$ 或 $\{3,4,8\}$ 代表之。

定义 3.2（交换群）　考虑 $(G,*)$，其中 G 为集合，而 $*$ 为运算。令公理：

（1）封闭性：$\forall x$、$y \in G$ 则 $x*y \in G$。

（2）交换律：$\forall x$、$y \in G$ 则 $x*y=y*x$。

（3）结合律：$\forall x$、y、$z \in G$ 则 $(x*y)*z=x*(y*z)$。

（4）存在单位素：$\forall x \in G$，$\exists e \in G$，使得 $x*e=x=e*x$。

（5）存在反元素：$\forall x \in G$，$\exists x' \in G$ 使得 $x*x'=e=x'*x$。

若公理 1、3、4、5 成立，称 $(G,*)$ 为群（Group）；若以上公理 1～5 都成立，称 $(G,*)$ 为交换群。

例 3.4　许多熟知的集合上都有群的代数结构，如整数及加法 $(\mathbb{Z},+)$、非零有理数及乘法 $(\mathbb{Q}\setminus\{0\},\cdot)$ 以及 $(\mathbb{F}_2,\oplus)$，其中 $\mathbb{F}_2=\{0,1\}$ 及 XOR 运算 $\oplus$ ($1\oplus 1=0$，$1\oplus 0=0\oplus 1=0$，$0\oplus 0=0$)等。这些都是交换群。

若再加上乘法，考虑 $(\mathbb{Z}/n,+,\cdot)$，此时除了 $(\mathbb{Z}/n,+)$ 为交换群以外，另外针对乘法运算也满足封闭性、交换律、结合律以及存在乘法单位素（即 $[1]=\{x\in\mathbb{Z} \mid x\equiv 1 \pmod n\}$）等性质，但并非所有非零元素都有乘法反元素，另外乘法对加法有分配律，即：

$$\text{若 } x、y、z \in \mathbb{Z}/n \text{ 则 } x\cdot(y+z)=x\cdot y+x\cdot z$$

此时，以代数的术语，称 $(\mathbb{Z}/n,+,\cdot)$ 为交换环（Commutative Ring）。注[2]而在讨论模运算的“除法”之前，就要了解何为同余类[a]在 $\mathbb{Z}/n$ 的“乘法反元素”，即解

$$ax\equiv 1 \pmod n。$$

在解此同余方程式之前，先复习辗转相除法（Euclidean Algorithm）：

例 3.5　求 7812 及 6084 的最大公因子 gcd(7812, 6084)。考虑

被除数=商×除数+余数，

gcd（被除数，除数）=gcd（除数，余数）

辗转相除法就是利用此性质，反复以

（除数/余数）取代（被除数/除数），

计算直至余数为 0，其计算过程如表 3-1 所示。

表 3-1　辗转相除过程

k	0	1	2	3	4	5	6	7
r_k	7812	6048	1728	900	828	72	36	0
q_k		1	3	1	1	11	2	

注[2] 可参考一般代数教科书，如 Serge Lang 的“Undergraduate Algebra” Second Edition，Springer Verlag。

其中，$r_k = q_{k+1}r_{k+1} + r_{k+2}$。所以gcd (7812 , 6084) = 36。

其 C 程序代码为：

程序 3.1 程序 gcd()用辗转相除法计算 gcd(a,b)：

```
unsigned int gcd(a,b) //a,b>0
{
    unsigned int r= a % b;
    if (r==0)
        return b;
    else
        return gcd(b,r);// （除数/余数）取代（被除数/除数）
}
```

我们在上例中，保留逐次所得的商 q_k，其主要目的是只要稍微修改辗转相除法，即可找出整数 x、y 使得

$$ax + by = gcd(a,b) 。$$

例 3.6（续上例） 求整数 x、y 使得 $7812x + 6084y = gcd = 36$。

其计算过程如 3-2 所示。

表 3-2 辗转相除过程

k	0	1	2	3	4	5	6	7
r_k	7812	6048	1728	900	828	72	36	0
q_k		1	3	1	1	11	2	
x_k	1	0	1	3	4	7	81	
y_k	0	1	1	4	5	9	104	

其中，

$$\begin{cases} x_{k+1} = q_k x_k + x_{k-1} \\ y_{k+1} = q_k y_k + y_{k-1} \end{cases} ,$$

而初始条件

$$\begin{cases} x_0 = 1 , x_1 = 0 \\ y_0 = 0 , y_1 = 1 \end{cases} 。$$

不难证明

$$r_k = (-1)^k x_k a + (-1)^{k+1} y_k b , \quad (k = 1,2,\cdots) 。$$

因此在本例中 $x = (-1)^6 81 = 81$, $y = (-1)^7 104 = -104$，得

$$81 \times 7812 + 6084 \times (-104) = 36 。$$

程序 3.2 xeuclidean()广义辗转相除法(Extended Euclidean Algorithm)：给定正整数 a、b，求整数 x、y 及 gcd 使得 $ax + by = gcd$。

```
struc xgcd{// 答案形态
    int x,y,gcd;
    };
int x[2],y[2];// 初始条件
```

```
x[0]=1;x[1]=0;
y[0]=0;y[1]=1;
xgcd xeuclidean(int a,int b) //a,b>0
{
    xgcd result;
    int q,r;
    int xx,yy;
    q=a/b;
    r=a%b;
    xx=x[1];
    yy=y[1];
    x[1]=-q*x[1]+x[0];
    y[1]=-q*y[1]+y[0];
    x[0]=xx;
    y[0]=yy;
    if (r==0)
    {// 答案
        result.gcd=b;
        result.x=x[0];
        result.y=y[0];
        return result;
    }
    else
        xeuclidean(b,r);//  （除数/余数）取代（被除数/除数）
};
```

有了 xeuclidean()，就可求 a 在 $\mathbb{Z}/n$ 的乘法反元素（Multiplicative Inverse）；若 $\gcd(a,n)>1$，即 a 与 n 不互质，此时乘法反元素不存在；但若 $\gcd(a,n)=1$，即 a 与 n 互质，此时乘法反元素即为 $ax+by=1$ 式中的 x 。考虑整数线性方程

$$ax+by=d$$

是否有整数解，这可由下列定理来回答。

定理 3.1　整数线性方程

$$ax+by=d$$

有整数解 ⇔

$$\gcd(a,b)\mid d \text{ 。}$$

证明：借助广义辗转相除法，存在整数 x_0、y_0，使得

$$ax_0+by_0=\gcd(a,b)\text{ 。}$$

(⇐) 若 $d=c\gcd(a,b)$，则

$$(x,y)=(cx_0,cy_0)$$

为一整数解。

(⇒) 若

$$ax+by=d$$

有整数解，因 $\gcd(a,b)\mid a$，且 $\gcd(a,b)\mid b$，故 $\gcd(a,b)\mid d$ 。

例 3.7 计算 $3x \equiv 1 \pmod{26}$：

运用 xeuclidean()计算可得如表 3-3 所示结果。

表 3-3 辗转相除过程

k	0	1	2	3	4	5
r_k	3	26	3	2	1	0
q_k	0	0	8	1	2	
x_k	1	0	1	8	9	

所以 $x = (-1)^4 x_k = 9$，$3^{-1} \pmod{26} \equiv 9 \pmod{26}$。

例 3.8 令 n 为自然数，则

$$\mathbb{Z}/n^{\times} = \{x \in \mathbb{Z}/n \mid x\text{有乘法反元素}\}$$
$$= \{x \in \mathbb{Z}/n \mid \gcd(x,n) = 1\}$$

对模乘法为交换群。

证明：交换群 5 大公理中，我们检查封闭性：即

$$x、w \in \mathbb{Z}/n^{\times} \text{ 则 } xw \in \mathbb{Z}/n^{\times}$$

因为 x、$w \in \mathbb{Z}/n^{\times}$，所以存在乘法反元素 x^{-1}、w^{-1} 使得

$$xx^{-1} \equiv 1 \text{ 且 } ww^{-1} \equiv 1,$$

而

$$(w^{-1}x^{-1})(xw) \equiv 1,$$

故 $w^{-1}x^{-1}$ 为 xw 的乘法反元素。

表 3-4 为常用模运算的计算复杂度。

表 3-4 模运算

运算		计算位复杂度
模加法	$a + b \pmod{n}$	$O(\log_2 n)$
模减法	$a - b \pmod{n}$	$O(\log_2 n)$
模乘法	$a \times b \pmod{n}$	$O((\log_2 n)^2)$
乘法反元素	$a^{-1} \pmod{n}$	$O((\log_2 n)^2)$

3.2 中国余式子定理（Chinese Remainder Theorem）

中国余式定理（Chinese Remainder Theorem），即俗称的韩信点兵，又称孙子点兵、鬼谷算、秦王暗点兵，大概在先秦战国时代，就已有记载。注[3]

例 3.9 “今有物不知其数，三三数之余二，五五数之余三，七七数之余二，问物几何？”

注[3] 可参见《辞海》中《孙子数物篇》。

有解箴诗：注[4]

“三人同行七十稀，五树梅花廿一枝，七子团圆正半月，除百零五便得知。”

令除数 $n_1=3$，$n_2=5$，$n_3=7$ 两两互质，$n=n_1n_2n_3=105$，令余数 $x_1=2$，$x_2=3$，$x_3=2$。再令 $M_1=n/n_1=105/3=35$，$M_2=n/n_2=105/5=21$，$M_3=n/n_3=105/7=15$，利用辗转相除法或直接计算乘法反元素得

$$s_1=M_1^{-1}\ (\text{mod}\ n_1)=35^{-1}\equiv 2\ (\text{mod}\ 3)$$
$$s_2=M_2^{-1}\ (\text{mod}\ n_2)=21^{-1}\equiv 1\ (\text{mod}\ 5)$$
$$s_3=M_3^{-1}\ (\text{mod}\ n_3)=15^{-1}\equiv 1\ (\text{mod}\ 7)$$

所以解为

$$\begin{aligned} x &\equiv x_1M_1s_1+x_2M_2s_2+x_3M_3s_3\quad (\text{mod}\ n)\\ &\equiv 2\times 70+3\times 21+2\times 15\\ &\equiv 23\ (\text{mod}\ 105)\ 。\end{aligned}$$

可将上列计算过程略为整理推广，便可写出如下的中国余式定理：

定理 3.2（中国余式定理，Chinese Remainder Theorem）　令 n_1、n_2、…、n_k 为两两互质的正整数，令 $n=n_1n_2\cdots n_k$。则同余联立方程组

$$\begin{cases} x\equiv x_1\ \ (\text{mod}\ n_1)\\ x\equiv x_2\ \ (\text{mod}\ n_2)\\ \qquad\vdots\\ x\equiv x_k\ \ (\text{mod}\ n_k)\end{cases}$$

在集合 $\{0,1,2,\cdots,n-1\}$ 有惟一解，其解为

$$x=\sum_{i=1}^{k}x_iM_is_i\ \ (\text{mod}\ n)$$

其中 $M_i=n/n_i$，而 $s_i\equiv M_i^{-1}\ \ (\text{mod}\ n_i)\ \ (i=0,1,\cdots,k)$。

此中国余式定理的证明除了考虑解的惟一性外，存在性的证明用定理中的公式也可，如同上例；而其应用面之广，也绝非只是点兵数物而已，至少在数学领域、信息领域均有非常重要的应用，在此举一简例加以说明。

例 3.10　解 $x^2\equiv 1\ (\text{mod}\ 63)$。已知 63=9×7，利用中国余式定理，即

$$x^2\equiv 1\ \ (\text{mod}\ 63)\Leftrightarrow\begin{cases} x^2\equiv 1\ \ (\text{mod}\ 7)\\ x^2\equiv 1\ \ (\text{mod}\ 9)\end{cases}$$

而 $x^2\equiv 1\ (\text{mod}\ 7)$ 有两个解，即 $x_1=\pm 1\ (\text{mod}\ 7)$；$x^2\equiv 1\ (\text{mod}\ 9)$ 也有两个解，即 $x_2=\pm 1\ (\text{mod}\ 9)$。利用中国余式定理解公式

$$x\equiv x_1M_1s_1+x_2M_2s_2\quad (\text{mod}\ n)\qquad 其中\ M_is_i\equiv 1\quad (\text{mod}\ n_i)$$

注[4] 程大位所做的诗。又见金庸著作《射雕英雄传》中，黄蓉云：“三人同行七十稀，五数梅花廿一枝，七子团圆正半月，余百零五便得知”。

$$\equiv (\pm1)9\times4+(\pm1)7\times4 \pmod{63}$$
$$\equiv 1,8,55,62 \pmod{63}\text{ 。}$$

中国余式定理其实有非常广义的陈述，假设整数 n 的质因数分解为

$$n=\prod_{i=1}^{k}p_i^{e_i}\text{ ,}$$

其中，p_i 为 n 的质因数，$e_i>0$，以乘法群 $\mathbb{Z}/n^{\times}$ 为例，就有

性质 3.1 存在群同构（Group Isomorphism）[注[5]]

$$f:\mathbb{Z}/n^{\times}\to\prod_{i=1}^{k}(\mathbb{Z}/p_i^{e_i})^{\times}\text{ 。}$$

定义 3.3（Euler-Phi 函数） 当 n 为正整数时，定义 Euler-Phi 函数为

$$\phi(n):=\left|\mathbb{Z}/n^{\times}\right|\text{ 。}$$

性质 3.2

$$\phi(n)=\prod_{i=1}^{k}\phi(p_i^{e_i})=n\prod_{i=1}^{k}\phi(1-\frac{1}{p_i})\text{ 。}$$

证明：由性质 3.1 与

$$\phi(p_i^{e_i})=p_i^{e_i}(1-\frac{1}{p_i})\text{ 得证。}$$

3.3 Lagrange 定理与费马小定理

在此我们继续深化群论的部分概念，这在作理论性的思考时，将有相当大的裨益。

定义 3.4（子群，Subgroup） 令 $(G,*)$ 为群，若 $H\subset G$ 为子集，且在相同的运算 $*$ 形成群则称 $(H,*)$（或 H）为 G 的子群（Subgroup）。

例 3.11 考虑乘法群 $G=\mathbb{Z}/5^{\times}$，此时 $G=\{[1],[2],[3],[4]\}$。令 $H=\{[1],[4]\}$，其中 $H\subset G$ 且满足群的各项公理，所以 H 为 G 的子群。

定理 3.3（Lagrange 定理） 若 G 为有限群，H 为 G 中之子群，则

$$\#H|\#G\text{ 。}$$

证明：H 为 G 的子群，为方便起见，假设为乘法群。可定等价关系如下：

$$\text{若 } a\text{、}b\in G,\ a\sim b\Leftrightarrow ab^{-1}\in H\text{ 。}$$

（读者请自行检查反身性、对称性及传递性三个性质）如此定义出的等价关系可将 G 分割成若干个等价类，即

$$[a]=\{ha\mid h\in H\},\ a\in G\text{ 。}$$

每个等价类都有 #H 个元素（考虑 $H\to[a]\equiv(h\mapsto ha)$ 为 1－1 对应）。因此 #H 整除 #G 。

注[5] 函数 f 群同构意味着：（1）f 是 1-1。

（2）f(x×y)=f(x)×f(y)。

在公开密钥密码系统中，往往会考虑模指数函数

$$f(x)=a^x \pmod p,$$

其中 p 为质数、a 为整数，此时函数 f(x) 是周期函数，其周期必然整除 $p-1$，这个重要性质可由费马小定理刻画。

定理 3.4（费马小定理，Fermat's Little Theorem）　令 p 为质数、a 为与 p 互质的整数，则[注 6]

$$a^{p-1}\equiv 1 \pmod p。$$

证明：考虑乘法群 $G=\mathbb{Z}/p^{\times}$，$H=<[a]>$ 为其子群，根据 Lagrange 定理，

$$n=\#H|\#G=p-1,$$

所以 $H=\{[1],[a],[a^2],\cdots,[a^n]\}$，其中 $a^n\equiv 1 \pmod p$。因此

$$a^{p-1}=(a^n)^{\frac{p-1}{n}}\equiv 1 \pmod p,$$

得证。

注意：事实上，若 a 为与 p 互质的整数，则 $\mathbb{Z}/p^{\times}=<[a]>$。

3.4　原根

考虑 2 的次方（mod 11）：

$$2^1\equiv 2, 2^2\equiv 4, 2^3\equiv 8, 2^4\equiv 5, 2^5\equiv 10,$$
$$2^6\equiv 9, 2^7\equiv 7, 2^8\equiv 3, 2^9\equiv 6, 2^{10}\equiv 1。$$

我们发现乘法群 $\mathbb{Z}/11^{\times}$ 中的同余类均可表示为[2]的若干次方，此时称2为乘法群 $\mathbb{Z}/11^{\times}$ 的原根（Primitive Root），同时仔细观察，当 $2^x\equiv 1$ 时，则10必整除 x；此时称10为2在（mod 11）（或在乘法群 $\mathbb{Z}/11^{\times}$）的秩（Order），详述如下。

定义 3.5　令 G 为乘法群，而 $g\in G$ 为其中一元素，则元素 g 的秩（Order）定义为

$$\mathrm{ord}(g):=\min\{x\in\mathbb{N}\mid g^x=1\}。$$

也可能不存在 $x\in\mathbb{N}$ 使得 $g^x=1$，此时定义 $\mathrm{ord}(g):=+\infty$。若 G 为有限群，则 $<g>:=\{g^x|x\in\mathbb{N}\}$ 为 G 的子群，有 $\mathrm{ord}(g)=\#<g>$，根据 Lagrange 定理，子群的元素个数必整除母群 G 的元素个数，故

$$\mathrm{ord}(g)|\#G。$$

例 3.12　在乘法群 $\mathbb{Z}/11^{\times}$ 中考虑所有元素的秩，如表 3-5 所示。

注[6] 费马（Pierre de Fermat，1601-1665）法国律师，历史上最有名的业余数学家，他的费马最后定理（Fermat's Last Theorem）直到上个世纪末，才由英国数论家 Andrew Wiles 得以论证。

费马定理（Fermat's Last Theorem）为：令整数 $n\geqslant 3$，则 $x^n+y^n=z^n$ 无非零整数解。

表 3-5 $\mathbb{Z}/11^{\times}$ 中所有元素的秩

g	1	2	3	4	5	6	7	8	9	10
ord(g)	1	10	5	5	5	10	10	10	10	2

其中秩为 10 的有 2、6、7、8、9 共 5 个，其实一般而言，在乘法群 $\mathbb{Z}/p^{\times}$（p 为质数）中有原根 $\phi(p-1)$ 个。而所有元素秩 ord(g) 都整除 $10=\#\mathbb{Z}/11^{\times}$。

原根在公开密钥问题“离散对数”的计算上有其重要性，在此列举一重要性质。

定理 3.5 令 g 为质数 p 上的原根，则

（1）若 x 为整数，则 $g^x \equiv 1 \pmod p \Leftrightarrow x \equiv 0 \pmod{p-1}$。

（2）若 i、j 为整数，则 $g^i \equiv g^j \pmod p \Leftrightarrow i \equiv j \pmod{p-1}$。

证明：

（1）($\Rightarrow$) 若 $g^x \equiv 1 \pmod p$，欲证 $p-1|x$：先假设不对，可写成 $x=(p-1)y+r$，其中 $0<r<p-1$。得

$1 \equiv g^x \equiv (g^y)^{p-1} g^r \equiv 1 \cdot g^r \equiv g^r \pmod p$（费马小定理）

但 $g \equiv g^{r+1}$，$g \equiv g^{r+2}$，…，所以 $<g>:=\{g^z \mid z \in \mathbb{N}\}$ 有 r 个元素，这与 g 为原根的假设矛盾。故 $r=0$，x 被 $p-1$ 整除。

($\Leftarrow$) 直接利用费马小定理可得。

（2）假设 i>j，将同余式

$$g^i \equiv g^j \pmod p$$

两边乘以 $(g^{-1})^j$，得

$$g^{i-j} \equiv 1 \pmod p,$$

利用已证明的性质 1，上式等价于

$$i-j \equiv 0 \pmod{p-1} \Leftrightarrow i \equiv j \pmod{p-1},$$

得证。

注意：已经有 G 为群且 $H \subset G$，此时只需要检查封闭性及对每一元素 $h \in H$ 都存在反元素，就足以说明 H 为 G 的子群。

例 3.13 令 G 为任一乘法群，$g \in G$ 为任一元素，则 $<g>:=\{g^k \in \mathbb{Z}\}$ 为 G 中的子群（封闭性与反元素的存在性自然成立）。此子群 $<g>$ 称为由元素 g 所生成的子群。

定义 3.6（循环群，Cyclic Group） 若存在 $g \in G$ 使得 $<g>=G$，则称 G 为循环群（Cyclic Group），而 g 为原根或生成元（Generator）。

例 3.14 考虑乘法群 $G=\mathbb{Z}/8^{\times}$，$G=\{[1],[3],[5],[7]\}$，其中

$$<[1]>=\{[1]\},$$
$$<[3]>=\{[1],[3]\},$$
$$<[5]>=\{[1],[5]\},$$
$$<[7]>=\{[1],[7]\},$$

故 G 不为循环群，无原根。

3.5　二次剩余（Quadratic Residue）

二次剩余在密码学的计算中有其重要性，主要是考虑同余式

$$x^2 \equiv a \pmod{n}$$

是否有整数解，其中 a、n 为互质整数，若有解，称 a 为(mod n)的二次剩余(Quadratic Residue)，若无解则称 a 为（mod n）的非二次剩余（Quadratic Nonresidue）。

例 3.15　考虑乘法群 $\mathbb{Z}/11^{\times}$ 的二次剩余有

$$\{[x^2] \mid x \in \mathbb{Z}/11^{\times}\} = \{[1^2],[2^2],[3^2],[4^2],[5^2],[6^2],[7^2],[8^2],[9^2],[10^2]\}$$
$$= \{[1],[3],[5],[7],[9]\},$$

而非二次剩余的集合为 $\{[2],[6],[7],[8]\}$。

性质 3.2　令 p 为奇质数，可定义函数

$$f:\mathbb{Z}/p^{\times} \to \pm 1 \quad (a \mapsto a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}),$$

则

$$f(a)=1 \Leftrightarrow a \text{ 为二次剩余。}$$

其中

$$\#\{[a]\in \mathbb{Z}/p^{\times} \mid a \text{ 为二次剩余}\} = \#\{[a]\in \mathbb{Z}/p^{\times} \mid a \text{ 为非二次剩余}\} = \frac{p-1}{2}\text{。}$$

证明：

（1）先证明

$$\#\{[a]\in \mathbb{Z}/p^{\times} \mid a \text{ 为二次剩余}\} = \frac{p-1}{2}\text{。}$$

明显 $\{[1^2],[2^2],\cdots,[(\frac{p-1}{2})^2]\}$ 为所有的二次剩余（$i^2 \equiv (p-i)^2 \pmod{p}$）。

（2）证明 f 为完好定义（Well-defined）：由费马小定理得

$$a^{p-1}-1 \equiv 0 \pmod{p},$$

因 $p-1$ 为偶数，可将上式注[7]因式分解得

$$(a^{\frac{p-1}{2}}-1)(a^{\frac{p-1}{2}}+1) \equiv 0 \pmod{p},$$

因 p 为质数得 $a^{\frac{p-1}{2}-1} \equiv 0 \pmod{p}$ 或 $a^{\frac{p-1}{2}+1} \equiv 0 \pmod{p}$，即 $a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}$。

（3）$f(a)=1 \Leftrightarrow$ a 为二次剩余：若 a 为二次剩余，则 $x^2 \equiv a$ 有解，所以

$$a^{\frac{p-1}{2}} \equiv (x^2)^{\frac{p-1}{2}} \equiv x^{p-1} \equiv 1 \pmod{p} \qquad \text{（费马小定理），}$$

注[7]　我们在此用到代数性质多项式环 $Z/p[x] := \{f(x)=\sum_{i=0}^{n} a_i \mid a_i \in Z/p, n \in Z \geqslant 0\}$（其中 p 为质数）为唯一分解环（Unique Factorization Domain，UFD）的事实！读者注意：并非所有的多项式都有唯一分解的性质，如 $Z/4[x]$ 就无此性质。

$$x^2-1 \equiv (x-1)(x+1) \equiv (x-3)(x+3)$$

故 $a^{\frac{p-1}{2}} \equiv 1$ 至少有 $\frac{p-1}{2}$ 个解；但其次数（Degree）为 $\frac{p-1}{2}$，故恰有 $\frac{p-1}{2}$ 个解，其余的 $\frac{p-1}{2}$ 个非二次剩余，为 $a^{\frac{p-1}{2}} \equiv -1 \pmod p$ 的解。

基于此定理，可引入以下概念。

定义 3.7（Legendre 符号） 令 p 为质数，定义 Legendre 符号如下：

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & 若 p \mid a \\ 1 & 若 p \nmid a，a 是二次剩余 \\ -1 & 若 p \nmid a，a 不是二次剩余 \end{cases}$$

定理 3.5（Euler 判别） 令 p 为奇质数，a 与 p 互质。则

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \pmod p$$

证明：与性质 3.2 类似，在此从略。

基于 Euler 判别，读者可以自行验证下列结果：

性质 3.3 令 p 为奇质数，a、b 为与 p 互质的整数，则

（1）若 $a \equiv b \pmod p$，则 $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$。

（2）$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right)$。

（3）$\left(\frac{a^2}{p}\right) = 1$。

（4）$\left(\frac{-1}{p}\right) = \begin{cases} 1 & 若 p \equiv 1 \pmod 4 \\ -1 & 若 p \equiv 3 \pmod 4 \end{cases}$。

（5）$\left(\frac{2}{p}\right) = \begin{cases} 1 & 若 p \equiv \pm 1 \pmod 8 \\ -1 & 若 p \equiv \pm 3, \pm 5 \pmod 8 \end{cases}$。

定理 3.6（Quadratic Reciprocity） 令 p、q 为奇质数，则

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}。$$

注意：Quadratic Reciprocity 最早是由 Legendre 提出的，但第一个正确而完整的证明还是由高斯提出，此定理截至目前已提出超过 140 种不同的证明方法。

证明：高斯在他 18 岁那年提出了第一个正确而完整的证明如下：

当 $p \equiv q \equiv 3 \pmod 4$ 时，

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = -1，$$

其他的情况皆

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = 1。$$

德国犹太数学家 Carl Jacobi 推广了 Legendre 符号，如同 Legendre 符号一样，Jacobi 符号也满足类似的性质，也满足 Quadratic Reciprocity，这使得在计算 Legendre 符号时非常便利。

定义 3.8（Jacobi 符号）　令 a 为整数，$n>0$ 为奇整数，其质因数分解为

$$n=p_1^{e_1}p_2^{e_2}\cdots p_k^{e_k}\text{。}$$

定义 Jacobi 符号

$$\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\cdots\left(\frac{a}{p_k}\right)^{e_k}\text{。}$$

要注意

$$x^2\equiv a\pmod n\ \text{有解}\Rightarrow\left(\frac{a}{n}\right)=1\text{，}$$

但

$$x^2\equiv a\pmod n\ \text{有解}\not\Leftarrow\left(\frac{a}{n}\right)=1\text{，}$$

例如

$$\left(\frac{2}{15}\right)=\left(\frac{2}{3}\right)\left(\frac{2}{5}\right)=(-1)(-1)=1\text{，}$$

而 $x^2\equiv 2\pmod{15}$ 无解，这与 Legendre 符号不同：

$$x^2\equiv a\pmod p\ \text{有解}\Leftrightarrow\left(\frac{a}{n}\right)=1\text{。}$$

Jacobi 符号的重要性质整理如下。

性质 3.4　令 a、b、n 为整数，则

（1）当 $n>0$ 为奇整数，Jacobi 符号 $\left(\frac{a}{n}\right)$ 才可能有意义。

（2）若 n 为奇整数，

$$a\equiv b\pmod n\Rightarrow\left(\frac{a}{n}\right)=\left(\frac{b}{n}\right)\text{。}$$

（3）若 n 为奇整数，

$$\left(\frac{ab}{n}\right)=\left(\frac{a}{n}\right)\left(\frac{b}{n}\right)\text{。}$$

（4）若 n 为奇整数，

$$\left(\frac{-1}{n}\right)=\begin{cases}1 & \text{若}n\equiv 1\pmod 4\\ -1 & \text{若}n\equiv 3\pmod 4\end{cases}\text{。}$$

（5）若 n 为奇整数，

$$\left(\frac{2}{n}\right)=\begin{cases}1 & \text{若}n\equiv \pm1\pmod 8\\ -1 & \text{若}n\equiv \pm3,\pm5\pmod 8\end{cases}\text{。}$$

（6）Jacobi 符号的 Quadratic Reciprocity：

当 m、n 为奇整数时，

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right)=(-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}\text{。}$$

可将以上诸性质整理成计算 Jacobi 符号的程序代码。

程序 3.2 计算 Jacobi 符号 $\left(\frac{a}{n}\right)$：

```
int jacobi(int a,int n)
{
    if (n%2==0) throw Undefined();
    if (a>=n) a=a%n;//性质2
    int gcd=gcd(a,n);
    if (a==gcd)
        return 0;
    else if (gcd!=0)
        return jacobi(gcd,n)*jacobi(a/gcd,n);
    //性质3
if (a<0)//性质3，4
    if (n%4==1)
        return jacobi(-a,n);
    else
        return -jacobi(-a,n);
    while (a%2==0) //性质3,5
        if (n%8==1||n%8==7)
            return jacobi(a/2,n);
        else
            return -jacobi(a/2,n);
        if (((a-1)*(n-1)/4)%2==0)//性质6
            return jacobi(n,a);
    else
        return -jacobi(n,a);
}
```

注意：此程序的计算复杂度为 $O((\log_2 n)^2)$ 位计算。

例 3.16 $x^2\equiv 43 \pmod{101}$ 有解，计算

$$\begin{aligned}
\left(\frac{43}{101}\right)&=(-1)^{\frac{100\times 42}{4}}\left(\frac{101}{43}\right) && \text{(Quadratic Reciprocity)}\\
&=\left(\frac{15}{43}\right)\\
&=(-1)^{\frac{14\times 42}{4}}\left(\frac{43}{15}\right) && \text{(Quadratic Reciprocity)}\\
&=-\left(\frac{-2}{15}\right)\\
&=-1\cdot\left(\frac{-2}{3}\right)\left(\frac{-2}{5}\right)
\end{aligned}$$

$$= -1 \cdot \left(\frac{1}{3}\right)\left(\frac{3}{5}\right)$$
$$= -1 \cdot 1 \cdot (3^{\frac{5-1}{2}} \pmod 5) \qquad \text{(Euler 判别)}$$
$$= -1 \cdot 1 \cdot -1$$
$$=1 。$$

3.6　Galois 域

域（Field）为代数基本结构，中国台湾地区的翻译为“体”，应源自德文、法文，而中国内地则翻译成“域”，取 Field 的字义。

定义 3.9（域，Field）：令 K 为一集合，并含有两个运算“田”及“*”，则 (K, 田 ,*) 为域，$\Leftrightarrow$ 下列公理均成立。

- (K, 田)为交换群，即

（1）（田-封闭性）：$\forall x、y \in K \Rightarrow x 田 y \in K$ 。

（2）（田-单位素）：$\exists 0 \in K \Rightarrow \forall x \in K$：$x 田 0 = x = 0 田 x$。

（3）（田-反元素）：$\forall x \in K$，$\exists -x \in K \Rightarrow x 田 (-x) = 0 = (-x) 田 x$。

（4）（田-结合律）：$\forall x、y、z \in K \Rightarrow (x 田 y) 田 z = x 田 (y 田 z)$。

（5）（田-交换性）：$\forall x、y \in K \Rightarrow x 田 y = y 田 x$。

- $(K^*,*)$ 为交换群（$K^* = K \setminus \{0\}$），即。

（1）（*-封闭性）：$\forall x、y \in K^* \Rightarrow x * y \in K$ 。

（2）（*-单位素）：$\exists e \in K^* \Rightarrow \forall x \in K^*$：$x * e = x = e * x$ 。

（3）（*-反元素）：$\forall x \in K^*$，$\exists x^{-1} \in K^* \Rightarrow x * x^{-1} = e = x^{-1} * x$ 。

（4）（*-结合律）：$\forall x、y、z \in K^* \Rightarrow (x * y) * z = x * (y * z)$ 。

（5）（*-交换性）：$\forall x、y \in K^* \Rightarrow x * y = y * x$ 。

- * 对 ⊕ 有分配律：

$$\forall x、y、z \in K \Rightarrow x * (y 田 z) = (x * y) 田 (x * z) 。$$

数系中的有理数 $(\mathbb{Q},+,\cdot)$ 、实数 $(\mathbb{R},+,\cdot)$ 以及复数 $(\mathbb{C},+,\cdot)$ 均为域。但整数 $(\mathbb{Z},+,\cdot)$ 不为域，因为乘法反元素不普遍存在于 $\mathbb{Z}$ 中，至于自然数 $(\mathbb{N},+,\cdot)$ 连对加法运算都不是群，更不用说是域了。

例 3.17　前面所叙述的集合

$$\mathbb{Z}/p = \{1,2,3,\cdots,p-1\} \quad （p 为质数）$$

在模加法“+”及模乘法“×”

$$x + y := x + y \pmod p \text{ 及 } x \times y := xy \pmod p$$

是域。

若 (K, 田 ,*) 为域，且 K 为有限集合，则称之为有限域（Finite Field）或Galois域（Galois Field）。然而在AES中所用到的Galois域 $\mathbb{F}_{2^8}$，并非直接从整数 $\mathbb{Z}$ 做模运算可得。其中一个典型做法是先考虑多项式环

$$\mathbb{F}_2[t] := \{f(t) = \sum_{i=0}^{n} a_i t^i \mid 某n \in \mathbb{N}, a_i \in \mathbb{F}_2\},$$

再作模运算

$$(\text{mod } t^8 + t^4 + t^3 + t + 1),$$

其中$t^8 + t^4 + t^3 + t + 1$为首项系数为1的不可约（Monic Irreducible）$\mathbb{F}_2$多项式。值得注意的是，$\mathbb{F}_p[t]$（p为质数）与$\mathbb{Z}$一样都可进行除法、辗转相除法（Euclidean Algorithm）的注[8]，实际上考虑两多项式除法，$p(t)$、$m(t) \in \mathbb{F}_p[t]$可惟一表示成

$$p(t) = m(t)q(t) + r(t),$$

其中$q(t)$为商式，$r(t)$为余式（$\deg(r(t)) < \deg(m(t))$）。

例 3.18 令

$$p(t) = t^4 + 2t^2 + 1, \quad m(t) = t^2 + t + 1 \in \mathbb{F}_3[t]。$$

用“长”除法求$p(t)$除以$m(t)$，为方便起见可分离系数：

```
                         1+   2+   2
                   ----------------------
1   +1   +1  /  1+   0+   2+   0+   1
                1+   1+   1
                --------------
                     2+   1+   0
                     2+   2+   2
                     ----------------
                          2+   1+   1
                          2+   2+   2
                          -----------
                               2+   2
```

得

$$商\ q(t) = t^2 + 2t + 2、余式\ r(t) = 2t + 2;$$

注意，多项式的系数为$\mathbb{F}_3 = \mathbb{Z}/3$，所有的系数运算都是(mod 3)运算。

与$\mathbb{F}_p$由$\mathbb{Z}$做模运算(mod p)所得类似，

$$\mathbb{F}_p[t]/(l(t)) \quad （其中l(t)为首项系数为1的不可约多项式）$$

代表所有$\mathbb{F}_p$多项式模$l(t)$的同余类所形成的集合；由于$l(t)$为不可约多项式，所有非$l(t)$倍数的多项式，都与$l(t)$互质，下例说明如何利用多项式版广义辗转相除法，求乘法反元素。

例 3.19 令 Galois 体K定义为

$$K = \mathbb{F}_3[t]/(t^2 + 1),$$

（当中$t^2 + 1$为不可约$\mathbb{F}_3$的多项式）求t在K的乘法反元素t^{-1}。

运算过程如表 3-6 所示。

注[8] 代数的术语称之为 Euclidean Domain。

表 3-6 运算过程

k	0	1	2	3	
r_k	t	t^2+1	t	1	0
q_k		0	t	t	
x_k	1	0	1	t	
y_k	0	1	0	1	

其中

$$\begin{cases} x_{k+1} = q_k x_k + x_{k-1} \\ y_{k+1} = q_k y_k + y_{k-1} \end{cases},$$

初始条件：

$$\begin{aligned} & x_0 = 1, x_1 = 0, y_0 = 0, y_1 = 1 , \\ & \Rightarrow r_k = (-1)^k x_k r_0 + (-1)^{k+1} y_k r_1 \\ & \Rightarrow (-t)(t) + (t^2+1) = 1 \\ & \Rightarrow (-t)(t) \equiv 1 \pmod{t^2+1} \\ & \Rightarrow t^{-1} \equiv -t \pmod{t^2+1} \end{aligned}$$

而 $\#K = 3^2 = 9$ 。

值得注意，所有的 Galois 域，凡是其元素个数（又称为秩）相同，皆同构，故

$$\mathbb{F}_3[t]/(t^2+1) \cong \mathbb{F}_3[t]/(t^2+t+1),$$

（t^2+1， t^2+t+1 皆为次数 3 的不可约多项式）因此在考虑 Calois 域 K 的代数结构时，只需知道 K 的秩，任何选取的模型都是同构，事实上 Galois 域 K 的秩必然是 $\#K = p^n$（其中 p 为质数， n 为正整数）；在考虑 K 上的计算时，一般是考虑模型 $K \cong \mathbb{F}_p[t]/(l(t))$，（$l(t)$ 为首项系数为 1、次数为 n 的不可约的 $\mathbb{F}_p$ 的多项式）。在此比较 Galois 域 $\mathbb{F}_p$ 与 $\mathbb{F}_p[t]/(l(t))$ 的类似之处：

$$\mathbb{Z} \leftrightarrow \mathbb{F}_p[t]$$

质数 q ↔ 首项系数为 1、次数为 n 的不可约多项式 $l(t)$ 。

$$(\bmod\ q) \leftrightarrow (\bmod\ l(t))$$

$$|\bullet| \leftrightarrow \deg(\bullet)$$

$$\mathbb{Z}/q \leftrightarrow \mathbb{F}_p[t]/(l(t))$$

$$\mathbb{F}_q \leftrightarrow \mathbb{F}_{p^n}$$

AES 所用的 Galois 域 $\mathbb{F}_{2^8} \cong \mathbb{F}_2[t]/(t^8+t^4+t^3+t+1)$ 的元素可很自然地化为二进位数，譬如

$$t^7+t^6+t^3+t+1 \equiv (11001011)_2,$$

而 $\mathbb{F}_2$ 多项式的加法，即 XOR 运算，如

$$\begin{aligned} & (t^7+t^6+t^3+t+1)+(t^7+t^5+t^2) \\ & = t^6+t^5+t^3+t^2+t+1 \equiv (01101111)_2 \\ & = (11001011)_2 \boxplus (10100100)_2 , \end{aligned}$$

而乘以t，即左移一位，末位填0，如

$$(t^5+t^3+1)t=t^6+t^4+t\equiv(01010010)_2=(00101001)_2\ll 1,$$

这些计算，包括模运算，都非常容易以硬件实现。另外值得一提的则是 $\mathbb{F}_{p^n}^{\times}=\mathbb{F}_{p^n}\setminus\{0\}$ 乘法群仍为一循环群，因此所有以离散对数（见本书“非对称密钥密码系统与离散对数”一章）为主的密码系统，均可推广为 $\mathbb{F}_{p^n}^{\times}$ 问题。

3.7 质数理论

本书所探讨的所有公开密钥密码都需要质数。质数问题本来只是少数数学家关心的问题，而其中所用到的工具，除了少数精通数论工具的数学家能够有较深刻的认识外，绝非一般大学数学课程所能涵盖。然而随着公开密钥密码系统的普遍使用，使得与质数相关的理论越发重要，详细介绍质数相关理论不是本书的要点，在此仅将应用较为普遍的质数知识做一介绍。

定义 3.10 令 p 为不为 1 的正整数，p 为质数(Prime)⇔若某正整数 d 整除 p(记为 $d\mid p$)，则 $d=1$ 或 $d=p$。

定理 3.7（Euclid） 存在无穷多个质数。

证明：采用反证法证明。假设仅存在有限多个质数，即 p_1、p_2、p_3、…、p_k 共k个。令

$$p_{k+1}=p_1p_2\cdots p_k+1$$

为一正整数，当中所有的质数 p_i（$i=1,2,3,\cdots,k$）皆无法整除 p_{k+1}，故数 p_{k+1} 也为质数，此与假设矛盾，故质数必有无限多个。

例 3.20 试求所有小于 200 的质数。

- 先将 2 至 199 规则列一个表。
- 2 是第一个质数，将表中所有其他 2 的倍数“筛”去，剩下

  3,5,7,9,11,13,15,17,19,21…。
- 表中下一个未“筛”去的数为质数 3，将表中所有其他 3 的倍数“筛”去，剩下

  5,7,11,13,17,19,23,25,29…。
- 表中下一个未“筛”去的数为质数 5，将表中所有其他 5 的倍数“筛”去。

  ⋮
- 一直“筛”到质数 $13<[\sqrt{199}]=14$ 的倍数为止，剩下的即所有小于 200 的 46 个质数：

2,	3,	5,	7,	11,	13,	17,	19,	23,	29,
31,	37,	41,	43,	47,	53,	59,	61,	67,	71,
73,	79,	83,	89,	97,	101,	103,	107,	109,	113,
127,	131,	137,	139,	149,	151,	157,	163,	167,	173,
179,	181,	191,	193,	197,	199。				

上述方法称之为 Eratosthenes 筛法（Sieve of Eratosthenes），也是公元前古希腊的一种简易找质数的算法，如图 3-2 所示。

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~
~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~	41	~~42~~	43	~~44~~	~~45~~
~~46~~	47	~~48~~	~~49~~	~~50~~	~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~	71	~~72~~	73	~~74~~	~~75~~
~~76~~	~~77~~	~~78~~	79	~~80~~	~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~	101	~~102~~	103	~~104~~	~~105~~
~~106~~	107	~~108~~	109	~~110~~	~~111~~	~~112~~	113	~~114~~	~~115~~	~~116~~	~~117~~	~~118~~	~~119~~	~~120~~
~~121~~	~~122~~	~~123~~	~~124~~	~~125~~	~~126~~	127	~~128~~	~~129~~	~~130~~	131	~~132~~	~~133~~	~~134~~	~~135~~
~~136~~	137	~~138~~	139	~~140~~	~~141~~	~~142~~	~~143~~	~~144~~	~~145~~	~~146~~	~~147~~	~~148~~	149	~~150~~
151	~~152~~	~~153~~	~~154~~	~~155~~	~~156~~	157	~~158~~	~~159~~	~~160~~	~~161~~	~~162~~	163	~~164~~	~~165~~
~~166~~	167	~~168~~	~~169~~	~~170~~	~~171~~	~~172~~	173	~~174~~	~~175~~	~~176~~	~~177~~	~~178~~	179	~~180~~
181	~~182~~	~~183~~	~~184~~	~~185~~	~~186~~	~~187~~	~~188~~	~~189~~	~~190~~	191	~~192~~	193	~~194~~	~~195~~
~~196~~	197	~~198~~	199	~~200~~	~~201~~	~~202~~	~~203~~	~~204~~	~~205~~	~~206~~	~~207~~	~~208~~	~~209~~	~~210~~

图 3-2　Eratosthenes 筛法表

程序 3.3（Eratosthenes 筛法）：将小于等于 N 的质数输出：

```
//布尔矩阵初始值为真
static bool* isPrime=new bool[N+1];
int i,j,k;
int N_sqrt =int(sqrt(N));

for (i=2;i<=N;i++) isPrime=1;
for (i=2;i<=N_sqrt;i++)
{
    k=N/i;
    if (isPrime[i])
    {
        for (j=2;j<=k;j++)
            isPrime[i*j]=0;
            //筛去 i*j 即 i 的倍数
    };
};
for (j=2;j<=N;j++)
    if (isPrime[j]==1)
        output<<j<<"\n";
        //j 为质数
```

数学王子高斯对于质数的分布有更深入的观察：令

$$\pi(x) := \#\{p \text{ 为质数} \mid p \leqslant x\}。$$

最初他猜测

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln(n)} = 1$$

（这对粗略的估算已经足够），稍后提出更为接近事实的猜测，后来被证明为真实，即

定理 3.8（质数定理，Prime Number Theorem）

$$\pi(n) \approx \mathrm{Li}(n) := \int_2^n \frac{dx}{\ln(x)}\text{。}$$

如图 3-3 所示。

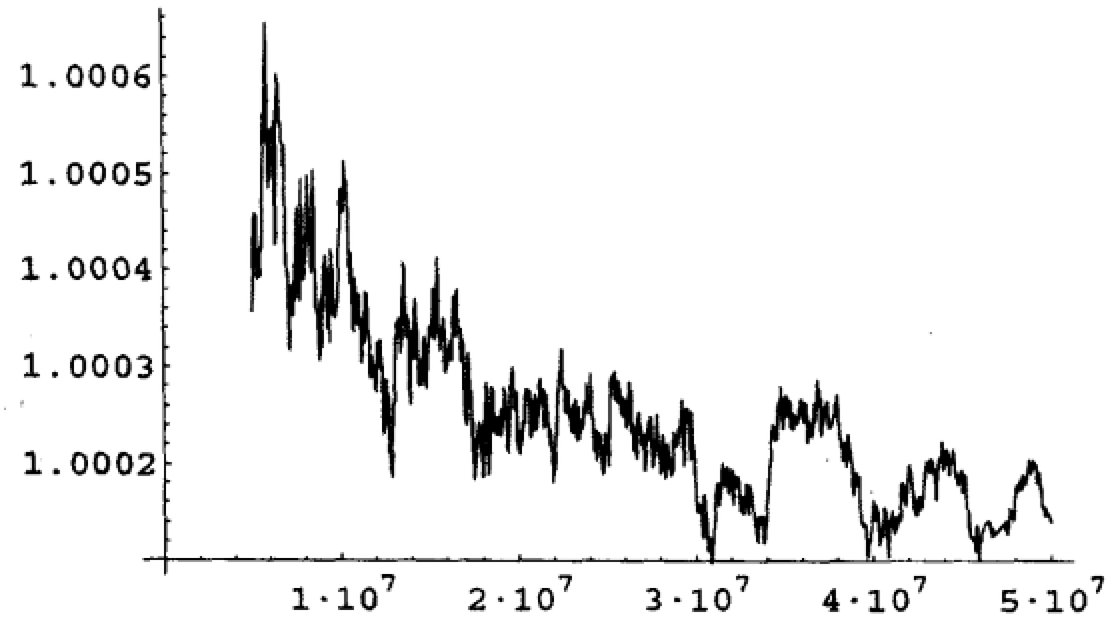


图 3-3 Li(n)/π(n)的比值

由此事实可见，随意取一整数 n，其为质数的概率是 $\frac{1}{\ln(n)}$，特别是对于固定位数的大质数的选取，也不怕会有不存在的现象。而熟知积分技巧的读者，也可试着利用分部积分（Integration by Parts）证明下式：

$$\mathrm{Li}(n) = \left.\frac{x}{\ln x} + \frac{x}{(\ln x)^2} + \frac{2x}{(\ln x)^3} + \cdots + \frac{k!x}{(\ln x)^{k+1}}\right|_2^n$$

$$+ (k+1)!\int_2^n \frac{x}{(\ln x)^{k+2}}dx \text{。}$$

高斯的弟子 Riemann，对于质数的分布给出了更为精确的观察：

$$\pi(x) \approx R(x) := \mathrm{Li}(x) - \frac{1}{2}\mathrm{Li}(\sqrt{x}) - \frac{1}{3}\mathrm{Li}(\sqrt[3]{x}) - \frac{1}{4}\mathrm{Li}(\sqrt[4]{x})\cdots\text{。}$$

假设大部分数学家都相信 Riemann 猜想是正确的，则

$$|\pi(x) - \mathrm{Li}(x)| = O(\sqrt{x}\log x)\text{。}$$

例 3.21

$$x = 10^8\text{，}\ \pi(x) = 5761455\text{，}\ R(x) = 5761552$$

$$x = 5\times10^8\text{，}\ \pi(x) = 26355867\text{，}\ R(x) = 26355517$$

$$x = 10^9\text{，}\ \pi(x) = 50847534\text{，}\ R(x) = 50847455$$

另外以离散对数问题为基础的公开密钥密码系统会考虑以下的质数问题：

“q 与 2q+1 同时为质数的可能性”。

一般称这样的质数为 Germain 质数，由 20 世纪数论大师 Hardy 与 Littlewood 于 1922 年提

出相当精确的猜想：

猜想 3.1（Hardy-Littlewood 猜想）

$$\#\{q \leqslant x \mid q 与 2q+1 同时为质数\} \approx \frac{2Cx}{\ln^2(x)}$$

其中 $C = 0.6601618158\cdots$ 如图 3-4 所示。

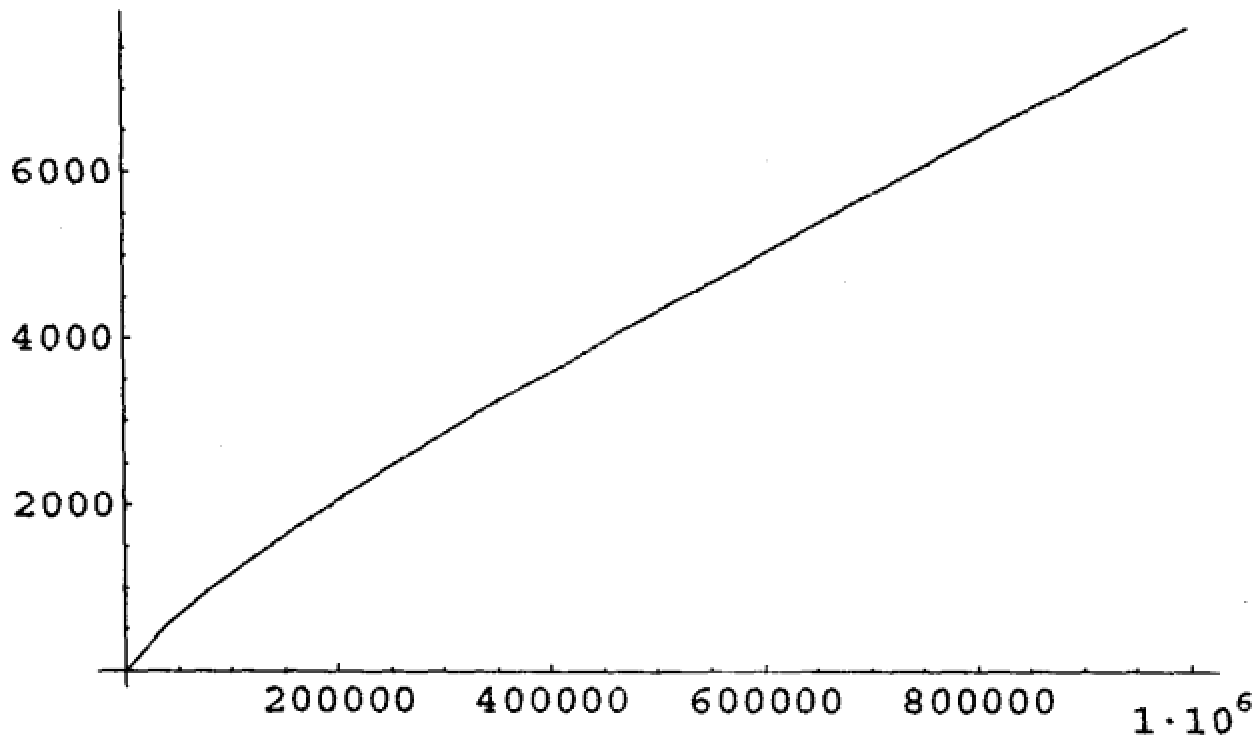


图 3-4　函数 $y = f(x) = \#\{q \leqslant x \mid q 与 2q+1 同时为质数\}$

例 3.22　前 12 对 Germain 质数为：

(2,5),　(3,7),　(5,11),　(11,23),
(23,47),　(29,59),　(41,83),　(53,107),
(83,167),　(89,179),　(113,227),　(131,263)。

另外对于找质数的算法，如 Rabin-Miller 算法、最近轰动信息界与数学界的 Agrawal-Kayal-Saxena 算法，也将在后面相关章节介绍。

3.8　连分数

连分数（Continued Fraction）是在实数范围内的一种表达数字的方式，在数论中常当作工具。所谓连分数，顾名思义，就是可以“多层”分数来表达的数，一般而言，所有的实数都可以表示为有限项或无限项的连分数。

定义 3.11（连分数）　任何以下形式的数均称为连分数

$$q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{q_4 + \cfrac{1}{q_5 + \ddots}}}},$$

其中，q_0、$q_1\cdots$ 为整数。

诚然，任何有理数$\frac{p}{q}$均可表示为有限项的连分数，其实做法就是利用辗转相除法，将每次除法所得的商逐次写下。

例 3.23 将有理数$\frac{147}{53}$化为连分数的形态，利用辗转相除法得如表 3-7 所示结果。

表 3-7 辗转相除结果

k	0	1	2	3	4	5	6	7
r_k	147	53	41	12	5	2	1	0
q_k		2	1	3	2	2	2	

故得

$$\frac{147}{53}=2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2}}}}}。$$

程序 3.4 给定x、y∈**N**，计算x/y的连分数

$$\frac{x}{y}=q[1]+\cfrac{1}{q[2]+\cfrac{1}{q[3]+\cfrac{1}{q[4]+\cfrac{1}{q[5]+\ddots+\cfrac{1}{q[k]}}}}},$$

其中，q[0]存k值。

```
int * ContinuedFraction(int x,int y)  // x,y>0
{
    int qq=x/y;
    int r=x-y*qq;
    int kmax=int(ceil(log(r)/log((1+sqrt(5))/2))+2);
    //最多 k 次除法,k<=log(r)/log((1+sqrt(5))/2))+2
    int * q=new int[kmax+1];
    q[1]=qq;
    int k=1;
    while(r!=0)
    {
        x=y;
        y=r;
        qq=x/y;
        q[++k]=qq;
        r=x-y*qq;
    };
    q[0]=k;//储存 k 值
    return q;
}
```

由于任何有限的连分数都是有理数，有时可利用连分数逼近估算，这在计算上也非常有用，以圆周率为例，有时我们会用 $\pi \approx 3.14159$ 估算，但也可利用分数 $\pi \approx \frac{22}{7}$ 估算，甚至用到更精确的有理数估算值 $\pi \approx \frac{333}{106}$，这些都可利用连分数说明。

例 3.24 列出 π 的连分数前若干项，并利用此方法估算 π，虽然 π 不是有理数，无法写成分数的形态，但依然可将辗转相除法略加修改，以符合问题的需求；注意其中

$$r_0 = \pi$$
$$r_1 = 1$$
$$q_k = [r_k]$$
$$r_{k+1} = \frac{1}{r_k - q_k}$$

r_k 不为整数，而 q_k 仍是整数，其计算结果如表 3-8 所示。

表 3-8 计算结果

k	r_k	q_k	c_k	d_k
0	π		1	0
1	1	3	3	1
2	0.1415926⋯	7	22	7
3	0.0625133⋯	15	333	106
4	0.09965945⋯	1	355	113
5	0.0034170⋯	292	103993	33102
⋮	⋮	⋮	⋮	⋮

其中递推关系为

$$\begin{cases} c_{k+1} = c_k q_{k+1} + c_{k-1} \\ d_{k+1} = d_k q_{k+1} + d_{k-1} \end{cases},$$

而初始条件为

$$\begin{cases} c_0 = 1, c_1 = q_1, \\ d_0 = 0, d_1 = 1, \end{cases}$$

这与辗转相除法xeuclidean()计算系数 x 、y 使得 $ax + by = \gcd(a,b)$ 的方法有不同之处，但原则上，仍是辗转相除法的推广。其中分数 $\frac{c_k}{d_k}$ 便是连分数中所给出的分数逼近值，即

$$3, \frac{22}{7}, \frac{333}{106}, \frac{355}{113}, \frac{103993}{33102}, \cdots。$$

由上例中，发现其实连分数的逼近速度很快，不超过 5 项，就可达 5、6 位的分数逼近。观察上例，可将各项系数整理出以下性质。

性质 3.5 令 $x \in \mathbb{R}$ 为一实数，其连分数表达式为

$$x = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \ddots}} ,$$

其中

$$r_0 = x$$
$$r_1 = 1$$
$$q_k = [r_k]$$
$$r_{k+1} = \frac{1}{r_k - q_k}$$

而其各项连分数的收敛值

$$\frac{c_k}{d_k} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{q_4 + \ddots + \cfrac{1}{q_k}}}} ,$$

当中 c_k 、d_k 满足递推关系及初始条件

$$\begin{cases} c_{k+1} = c_k q_{k+1} + c_{k-1} \\ d_{k+1} = d_k q_{k+1} + d_{k-1} \end{cases} ,$$

与

$$\begin{cases} c_0 = 1 , c_1 = q_1 , \\ d_0 = 0 , d_1 = 1 \end{cases} 。$$

其实第 k 项连分数收敛值 $\frac{c_k}{d_k}$ 与 x 的误差可由以下定理给定：

定理 3.9 令 $\frac{c}{d}$ （且 $\gcd(a,b) = 1$ ）为实数 x 的某项连分数的收敛值，⇔

$$\left| x - \frac{c}{d} \right| \leqslant \frac{1}{2d^2} 。$$

注意：证明可参考两位数论大师 G.H. Hardy 与 E.M. Wright 所著的 "*An Introduction to the Theory of Numbers*"。Oxford Clarendon Press, 4th edition (1975) 的 Theorem 177。

3.9 密码安全伪随机数生成器

任何近代的密码系统都需要好的 "伪随机数生成器"（Pseudom-Random Generator），用以产生随机数位。简易的伪随机数生成器，如 C 标准程序库中的 rand()，可产生 0～65535 之间的伪随机数整数，如此的伪随机函数，需要随机数种子（Seed）当作输入的初始值。一般而言，许多常用的伪随机数生成器，都是以线性同余生成器（Linear Congruential Generator）的方式产生数列 x_1 、x_2 、⋯ ，其中

$$x_n = ax_{n-1} + b \pmod{m} ,$$

其中 x_0 为随机数种子，如此的线性同余式是可预测的，用以产生 DES、AES、RSA 等近代密码系统的密钥的位，非常不合时宜；因为只要伪随机数生成器能轻易预测，即使有再好的密码算法，也是徒劳；即使伪随机数生成器是根据多项式同余生成器（Polynomial Congruential Generator），密码也是不安全的，这在使用密码系统时不可不慎重。

在密码学中，我们需要密码安全的伪随机数生成器，常用的方式有两类：

- 利用单向函数，如密码系统 DES 的加密函数、Hash 函数 SHA-1 等。定义
$$x_i = f(s+i)$$
其中 $i=1,2,3,\cdots$，s 为随机数种子，而 b_i 为 x_i 的最末位，则位数列 b_0，b_1，b_2，$\cdots$ 为不可预测的。
- 另外一种方式，就是利用数论的问题（这些与公开密钥密码算法较为相关）。

常用的一种伪随机数生成器，即 Blum-Blum-Shub 伪随机数生成器可用以下算法描述：

算法 3.1（Blum-Blum-Shub 伪随机数字生成器）　输出≪一个伪随机数数列 b_1，b_2，…，b_k。

```
BlumBlumShub()
{
    do
    {
        p=RandomPrime();
    }while (p%4!=3);
    do
    {
        q=RandomPrime();
    }while (p%4!=3);
    //p,q 为随机质数且 = 3 mod 4
    n=p*q;
    do{
        s=RandomInteger(1,n);
    }while(gcd(s,n)!=1);
    //gcd(s,n)=1 且 s 为随机数种子
    x[0]=s;
    for(i=1;i<=k;i++)
    {
        x[i]=x[i-1]*x[i-1]%n;
        b[i]=x[i]&1;//取最末位
    };
    return(b[1],b[2],...,b[k]);
}
```

注意：Blum-Blum-Shub 伪随机数生成器与公开密钥密码系统 Rabin 的密码算法（参见本书“Rabin 密码”一节）有类似之处，但 Blum-Blum-Shub 伪随机数生成器的计算缓慢，一种改良的方式是一次取 x[i] 的末 d 位，若 $d \le \log_2 \log_2 n$ 且 n 的因数分解难解，则都还是密码安全的。

若读者对公开密钥密码系统 RSA 有信心，也可考虑 RSA 伪随机数生成器：

算法 3.2（RSA 伪随机数字生成器）　输出≪一个伪随机数数列 b_1，b_2，…，b_k。

```
RSA_PseudomBitGen()
```

```
{
    p=RandomPrime();
    q=RandomPrime();
    n=p*q;
    phi=(p-1)*(q-1);
    do
    {
        e=RandomInteger(2,phi-1);
    }while(gcd(e,phi)!=1);
    //gcd(e,phi)=1
    x[0]=RandomInteger(1,n-1);
    //x[0]为随机数种子
    for(i=1;i<=k;i++)
    {
        x[i]=PowerMod(x[i-1],e,n);
        //x[i]=x[i-1]^e%n
        b[i]=x[i]&1;
        //取最末位
    };
    return(b[1],b[2],...,b[k]);
}
```

与 Blum-Blum-Shub 伪随机数生成器一样，也有一种改良的方式可加速产生随机数，即一次取 x[i] 的末 d 位，若 $d \leqslant \log_2 \log_2 n$ 且 n 的因数分解很困难，都算是密码安全的。

例 3.25 要以 RSA 伪随机数生成器产生一 9 位整数。

（1）随机取质数 $p = 28996819$ 与 $q = 60094427$，则

$$n = pq = 1742547222627713。$$

（2）随机取伪随机数种子 $x_0 = 58869063457997$，取 $e = 3$，代入递推式

$$x_i \equiv x_{i-1}^3 \pmod n，$$

得

$$\begin{aligned} x_1 &= 868757651707608 \\ x_2 &= 639725187266830 \\ x_3 &= 209736079624193 \\ x_4 &= 171142074174335 \\ x_5 &= 575410053225581 \\ x_6 &= 645887143191314 \\ x_7 &= 1722869328336461 \\ x_8 &= 112281758229122 \\ x_9 &= 1361742479237966。\end{aligned}$$

（3）由 $b_i \equiv x_i \pmod 2$ 得二元字串

$$b_1b_2b_3b_4b_5b_6b_7b_8b_9=(001110100)_2 = 116，$$

即为所得。

第4章　信息理论

信息理论（Information Theory）是信息学科中非常重要的理论基础之一，特别是与密码分析有紧密关联。信息理论的许多重要概念，是 Claude Shannon 在 1940 年代末期提出[注1]，前面所提及的"无条件安全"、"计算上安全"等概念，也是源自 Shannon 的论文，可以把信息理论当作工具，从而以量化分析的方式，分析密码系统的安全性。在 Shannon 的理论中，他提出所谓"熵"（Entropy）的概念，是用某个量化的数值来"测量"信息；"熵"还衍生出出自然语言熵（Entropy of the Natural Language）、重复率（Redundancy）以及 Unicity 距离等概念，这些概念对于当代密码分析有重要意义。在本章中也将说明为何单次密码本加密是无条件安全的。而在讨论这些问题之前，先复习一些重要的概率概念。

4.1　概率

概率是非常重要的应用数学科目，应用面很广，在此仅对必要的概念加以讨论。

定义 4.1　令 S 为一非空的有限集合，称为样本空间（Sample Space），其部分集合称为事件（Events）。在样本空间 S 上的概率分布（Probability Distribution）即用一个函数 p 将事件映射为某实数，

$$p : 2^S \mapsto \mathbb{R},$$

（其中 2^S 表示 S 的幂集合，即 $\{M \mid M \subset S\}$）满足下列各条件：

（1）$p(A) \geqslant 0$ 对所有的事件 $A \subset S$。

（2）$p(S) = 1$。

（3）$p(A \cup B) = p(A) + p(B)$，当两事件 A 与 B 互斥（即 $A \cap B = \varnothing$）。

若 A 为事件，则 $p(A)$ 为此事件的概率。

性质 4.1　可得简易性质：

符号同前，则

（1）$p(\varnothing) = 0$。

（2）若 $A \subset B$，则 $p(A) \leqslant p(B)$。

（3）$0 \leqslant p(A) \leqslant 1$，当 $A \subset S$。

（4）$p(S \setminus A) = 1 - p(A)$。

（5）若 A_1、A_2、…、A_n 均两两互斥，则

[注1] 可参阅 Claude Shannon 的两篇著名论文"A mathematical theory of communication" Bell Systems Technical Journal,27(1948)P.379-423,623-656 以及"Communication theory of secrecy systems" Bell Systems Technical Journal,28(1949)P.656-715。

$$p(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} p(A_i)\text{。}$$

（6） $p(A)=\sum_{a\in A} p(a)$ （其中 $p(a)=p(\{a\})$ ）。

定义 4.2（条件概率，Conditional Probability） 令 A 与 B 为事件，且 $p(B)\geqslant 0$ ，A 在条件 B 成立下的条件概率定义为

$$p(A\,|\,B)=\frac{p(A\cap B)}{p(B)}\text{。}$$

定义 4.3 两事件 A 与 B 称为独立事件（Independent Events）

$$\Leftrightarrow \quad p(A\cap B)=p(A)p(B)\text{。}$$

（此等式也等价于 $p(A\,|\,B)=p(A)$ ），若等式不成立，则称 A 与 B 为非独立事件（Dependent Events）。

定理 4.1（Bayes 定理） 若 A 与 B 为事件，且 $p(A)>0$， $p(B)>0$，则

$$p(B)p(A\,|\,B)=p(A)p(B\,|\,A)$$

证明：由条件概率的定义，得

$$p(A\,|\,B)p(B)=p(A\cap B)$$

与

$$p(B\,|\,A)p(A)=p(A\cap B)\text{。}$$

故得

$$p(B)p(A\,|\,B)=p(A)p(B\,|\,A)\text{。}$$

4.2 完美秘密

Shannon 曾引入完美秘密（Perfect Secrecy）的概念，考虑以下场景：Alice 使用某密码系统加密信息成密文传给 Bob，而 Eve 可截收任何由 Alice 传给 Bob 的密文，Eve 试着从密文中分析找出可能与明文相关联的任何信息。一个拥有完美秘密的密码系统，就是指 Eve 无法从这些密文读出任何有关明文的信息的系统。如何将“完美秘密”的概念量化？一种较为精确的方式就是引入概率的概念。

假设所考虑的密码系统为有限明文空间 $\mathbb{M}$ 、有限密文空间 $\mathbb{C}$ 以及有限密钥空间 $\mathbb{K}$ ，加密函数表示为 E_k， $k\in\mathbb{K}$ ，而解密函数表示为 D_k， $k\in\mathbb{K}$ 。令样本空间为

$$\mathbb{M}\times\mathbb{K}=\{(m,k)\,|\,m\in\mathbb{M},k\in\mathbb{K}\}\text{，}$$

在样本空间中的事件 $m\in\mathbb{M}$ ，即 $\{(m,k)\,|\,k\in\mathbb{K}\}$ ，而样本空间中的事件 $k\in\mathbb{K}$ 即 $\{(m,k)\,|\,m\in\mathbb{M}\}$ ，其概率分布函数当然必须满足

$$p(m)=p_{\mathbb{M}}(m)\text{，}$$

$$p(k)=p_{\mathbb{K}}(k)\text{，}$$

其中 $p_{\mathbb{M}}$、$p_{\mathbb{K}}$ 表示原先在明文空间以及密钥空间的概率分布函数，事件 p 与事件 k 为独立事件。而在密文空间 $\mathbb{C}$ 中的任何一密文 c，为样本空间 $\mathbb{M}\times\mathbb{K}$ 的部分集合

$$\{(m,k)\,|\,E_k(m)=c\}\text{，}$$

即所有加密后会成为密文 c 的所有明文/密钥的有序对 (m,k) 所形成的集合。

如果 Eve 已经知道 Alice 与 Bob 所使用的明文语言，那么 Eve 应该可以知道明文空间上的概率分布函数 p_M。如果 Eve 所截获的密文 c 可能的明文为 m 的概率，即条件概率

$$p(m|c)$$

不等于该明文出现的概率 $p(m)=p_M(m)$ 时，Eve 就可能从密文 c 中得到用于破译的信息；若两者的概率相同，即

$$p(m|c)=p(m)$$

时，对于密文分析无益。因此，Shannon 的完美秘密的定义如下。

定义 4.4　一个密码系统定义为完美秘密（Perfect Secrecy）

$\Leftrightarrow$ 所有给定密文出现的事件与所有特定明文出现的事件，皆是独立事件。

$\Leftrightarrow$ $p(m|c)=p(m)$，对所有明文 m 以及所有密文 c 皆成立。

以下是 Shannon 著名的定理：

定理 4.2（Shannon）　令 $\#\mathbb{C}=\#\mathbb{K}$（所有密文的可能数目等于所有密钥的可能数目）且任一明文 m 出现的概率均为正数，即 $p(m)>0$。此密码系统为完美秘密 $\Leftrightarrow$ 下列条件皆成立：

（1）在密钥空间上的概率分布函数 $p_{\mathbb{K}}$ 为均匀分布。

（2）对任一明文 m 以及任一密文 c，均恰好仅存在一把密钥 k 使得

$$E_k(m)=c。$$

证明：（$\Rightarrow$）假设该密码系统为完美秘密。

- 对任一明文 m 以及任一密文 c 均恰好仅存在一把密钥 k 使得

  $$E_k(m)=c：$$

  假设存在某明文 m 以及某密文 c，并无密钥 k 使得 $E_k(m)=c$，此时

  $$p(m)\neq p(m|c)=0 \text{（因为 } p(m)>0\text{）},$$

  与假设矛盾，因此每一对明文/密文 (m,c)，必有密钥 k 使得 $E_k(m)=c$；但密钥总数等于密文总数，这样的密钥 k 只有一把，即恰有一把密钥满足 $E_k(m)=c$。

- $p_{\mathbb{K}}$ 为均匀分布的：

  先固定某密文 c，对某明文 m，令 $k(m)$ 为密钥使得 $E_{k(m)}(m)=c$。由 Bayes 定理可得

  $$p(m|c)=\frac{p(c|m)p(m)}{p(c)}=\frac{p(k(m))p(m)}{p(c)}$$

  对所有的明文 m 均成立。因密码系统为完美秘密，得 $p(m|c)=p(m)$，因此 $p(k(m))=p(c)$，当中 c 是固定的，m 变动，$k(m)$ 也随之变动，所有密钥 k 均可写成 $k(m)$，因此所有密钥事件的概率均相同。

（$\Leftarrow$）假设密钥空间上的概率分布函数为均匀分布，且任一明文 m 以及任一密文 c，恰有一把密钥 $k=k(m,c)$ 使得 $E_k(m)=c$，则

$$p(m|c)=\frac{p(m)p(c|m)}{p(c)}=\frac{p(m)p(k(m,c))}{\sum_{m\in\mathbf{M}}p(m)p(k(m,c))},$$

（又用到 Bayes 定理）此时 $p(k(m,c))=\frac{1}{\#\mathbb{K}}$，因此分母为

$$\sum_{m\in\mathbf{M}}p(m)p(k(m,c))=\frac{\sum_{m\in\mathbf{M}}p(m)}{\#\mathbb{K}}=\frac{1}{\#\mathbb{K}}$$

代回上式得 $p(m\mid c)=p(m)$。

例 4.1（单次密码本） 单次密码本为完美秘密的加密算法。Alice 欲加密 n 位的明文 m，其中明文空间 $\mathbb{M}$、密文空间 $\mathbb{C}$ 以及密钥空间 $\mathbb{K}$ 满足

$$\#\mathbb{M}=\#\mathbb{C}=\#\mathbb{K}=2^n。$$

而任一可能密钥的概率为 $\frac{1}{2^n}$，故密钥空间上概率分布函数为均匀的。且对任一 n 位的明文 m 以及 n 位的密文 c，恰有一 n 位的密钥 k 使得

$$E_k(m)=m\oplus k=c\ (\Leftrightarrow k=m\oplus c)。$$

根据 Shannon 的定理，单次密码本为完美秘密。

4.3 熵

诚然，单次密码本是完美秘密的，但在实际的日常加密中却不实用，因为对于使用者、管理者而言，每次加密都用不同的密钥，成本过高，管理也不便，密钥的分配管理成了实际的问题。当代密码学中并不要求完美秘密的算法，而是寻找计算上安全的密码系统，并满足：

- 一把密钥可使用许多次。
- 一把短的密钥可以对长的信息加密。

这样的密码系统当然不可能无条件安全，这样，Eve 截收到 Alice 传送给 Bob 的密文，虽然可能找出一些明文/密文之间的信息关联性，但在某种程度上，她也无法确定密文的意义，即使截收到许多密文，甚至明文，也有很大的不确定性。Shannon 理论的“熵”就是“测量”信息间的不确定性。

定义 4.5（熵，Entropy） 令 A 为样本空间，X 为定义在样本空间上的随机变量，则随机变量 X 的熵定义为

$$H(X)=-\sum_{x\in A}p(X=x)\log_2(p(X=x))。$$

例 4.2 假设不公平地掷硬币。出现正面的概率为 p，而出现反面的概率为 $1-p$（$0<p<1$）。该事件的熵为

$$H(x)=-p\log_2 p-(1-p)\log_2(1-p),$$

$H(x)$ 为 p 的函数，其中 $H(x)$ 在 $p=\frac{1}{2}$ 达到极大值 1。

定义 4.6（连接熵，Joint Entropy） 令 X 与 Y 为样本空间 A 与 B 上的随机数，则连接熵 $H(X,Y)$ 定义为

$$H(X,Y)=-\sum_{(x,y)\in A\times B}p_{X\times Y}(x,y)\log_2(p_{X\times Y}(x,y))。$$

在密码系统中，在已知密文中，如果要“测量”密钥的不确定性，此时就要用到条件熵（Conditional Entropy）。

定义 4.7（条件熵，Conditional Entropy） 令 X 与 Y 为样本空间 A 与 B 上的随机数，则在条件 X 下的 Y 条件熵定义为

$$\begin{aligned}H(Y|X) &= \sum_{x\in A} p_X(x)H(Y|X=x)\\ &= -\sum_{x\in A} p_X(x)\left(\sum_{y\in B} p_Y(y|x)\log_2 p_Y(y|x)\right)\\ &= -\sum_{(x,y)\in A\times B} p_{X\times Y}(x,y)\log_2 p_Y(y|x)\text{。}\end{aligned}$$

上述不同熵的关联性，可以用下列定理描述。

定理 4.3（链式规则，Chain Rule）

$$H(X,Y)=H(X)+H(Y|X)\text{。}$$

证明：

$$\begin{aligned}H(X,Y) &= -\sum_{(x,y)\in A\times B} p_{X\times Y}(x,y)\log_2 p_{X\times Y}(x,y)\\ &= -\sum_{(x,y)\in A\times B} p_{X\times Y}(x,y)\log_2 p_X(x)p_Y(y|x) \qquad \text{（条件概率定义）}\\ &= -\sum_{(x,y)\in A\times B} p_{X\times Y}(x,y)\log_2 p_X(x)\\ &\quad -\sum_{(x,y)\in A\times B} p_{X\times Y}(x,y)\log_2 p_Y(y|x) \qquad \text{（指数律）}\\ &= -\left(\sum_{x\in A}\log_2 p_X(x)\sum_{y\in B} p_{X\times Y}(x,y)\right)+H(Y|X)\\ &= -\sum_{x\in A} p_X(x)\log_2 p_X(x)+H(Y|X)\\ &\quad (\because \sum_{y\in B} p_{X\times Y}(x,y)=p_X(x))\\ &= H(X)+H(Y|X)\end{aligned}$$

可利用链式规则以及概率分析得到以下性质：

性质 4.2

（1）$H(X)\leqslant \log 2|A|$，当样本空间 A 的各元素出现的概率相同时“=”成立。

（2）$H(X,Y)\leqslant H(X)+H(Y)$。

（3）$H(Y|X)\leqslant H(Y)$，当 X 与 Y 为独立事件时“=”成立。

我们也可利用熵来表示何时密码系统为完美秘密：

定理 4.4　令 M 为明文空间 M 的随机变量，C 为密文空间 $\mathbb{C}$ 的随机变量。密码系统为完美秘密 $\Leftrightarrow$ $H(M|C)=H(M)$。

注意：此处略去证明，读者可自行完成。由此定理以及上述性质可知，一个密码系统为完美秘密 $\Leftrightarrow$ 任何特定明文出现的事件与任何特定密文出现的事件为独立事件。尽管大多数的密码系统不满足此项性质的。

用熵的理论描述密码系统，就有以下简易结果：

- $H(M|K,C)=0$：熵值为 0 的意义就是没有不确定性。密码系统在密文、密钥已知的情况下，当然能知道密文，否则如何解密？

- $H(C|M,K)=0$：一个密码在明文、密钥已知的状况下，能加密得出密文。

条件熵 $H(K|C)$ 称为密钥模糊度（Key Equivocation），用以测量由密文所泄漏密钥信息的程度。

定理 4.5

$$H(K|C)=H(K)+H(M)-H(C)\text{。}$$

证明：

$$\begin{aligned}H(K,M,C)&=H(M,K)+H(C|M,K) \quad \text{（链式规则）}\\&=H(M,K) \quad (\because H(C|M,K)=0)\\&=H(K)+H(M) \quad \text{（K 与 M 为独立事件），}\end{aligned}$$

另外

$$\begin{aligned}H(K,M,C)&=H(K,C)+H(M|K,C) \quad \text{（链式规则）}\\&=H(K,C) \quad (\because H(M|K,C)=0)\\&=H(C)+H(K|C)\text{，}\end{aligned}$$

故由链式规则

$$H(K|C)=H(K,C)-H(C)=H(K)+H(M)-H(C)\text{。}$$

4.4 自然语言之熵

密码分析的最主要目的，就是要找出密钥，以密文攻击（Cipher-Only Attack）为例，假设攻击者 Eve 拥有无限的计算资源，并且可截收任何由 Alice 传给 Bob 的密文，若 Alice 的明文是某种“自然语言”，如英文、中文或是符合某种特定语法的语言，Eve 就可借助对该自然语言的信息，即自然语言的熵的了解，缩小密钥可能搜寻的范围，丢弃“假密钥”(Spurious Key)，因此对所谓假密钥总数的估计非常重要。

定义 4.8（假密钥，Spurious Key） 令 $c\in\mathbb{C}^n$ 为含 n 个字元的某密文。令

$$\mathbb{K}(c):=\{k\,|\,E_k(m)=c\text{，其中 m 为符合语法，有意义的明文}\}$$

为产生密文 c 的假密钥的集合。

以自然语言英文为例，共有明文字母 26 个，若每个字母平均出现，其熵为 $\log_2(26)\approx 4.70$，但实际情形并非如此（如字母 a 出现频率约为 0.82、字母 b 出现频率约为 0.15 等），因此其熵值应估计为

$$-(0.82\log_2 0.82+0.15\log_2 0.15+\cdots)\approx 4.18<4.70\text{。}$$

但是英文各字母的出现频率概率并非独立事件，必须进一步计算估计长度为 n 的不同字母排列出现频率概率的熵值。

例 4.3 在破译例 2.7 的密文过程中，假设已确定：

密文：	MTN	PX	YBGG	RPCY	JPX	BTJXVQVXJMJBCT.
明文：	and	he	-ill	sho-	the	inter-retation.

即使不再进一步分析，几乎可确定：

密文：	MTN	PX	YBGG	RPCY	JPX	BTJXVQVXJMJBCT.
明文：	and	he	will	show	the	interpretation.

而其不确定性，是微乎其微的。

自然语言熵就是“测量”自然语言字母排列的不确定性，其定义如下：

定义 4.9（自然语言熵，Entropy of the Natural Language）　令 $\mathbb{M}$ 为某自然语言的字母集合，$\mathbb{M}^n$ 表示长度为 n 的各种不同字母的排列。该自然语言 L 的熵值定义为

$$H_L = \lim_{n\to\infty}\frac{H(\mathbb{M}^n)}{n}。$$

此处熵值 H_L 就是该自然语言的熵值；而由此 $\mathbb{M}$ 中字母所产生的随机信息的熵是 $\log_2|\mathbb{M}|$，该自然语言的“重复率”（Redundancy）可定义为：

定义 4.10

$$R = 1-\frac{H_L}{\log_2|\mathbb{M}|}。$$

例 4.4　以英文为例，其自然语言熵一般估计为

$$1 \le H_L \le 1.5,$$

其语言重复率均为

$$R \approx 1-\frac{1.25}{\log_2 26} \approx 0.75。$$

这表示纯文字的 1MB 英文文件可压缩至 0.25MB。

而对于不同的加密方式，我们究竟要收集多少密文，才能有效破译？更明确地说，需要多长的密文，才能惟一决定其密钥值？

定义 4.11　Unicity 距离 n_0，就是该密码系统所产生密文而惟一决定惟一密钥值的密文长度。

Unicity 距离可由以下算式估计：

定理 4.6　Unicity 距离 n_0 估计为

$$n_0 \approx \frac{\log_2|\mathbb{K}|}{R\log_2|\mathbb{M}|}。$$

其中 $|\mathbb{K}|$ 表示所有可能的密钥数，而 $|\mathbb{M}|$ 表示所有的字母总数，R 即重复率。

证明：令 $c\in\mathbb{C}^n$ 为含 n 个字元的某密文。令

$$\mathbb{K}(c) := \{k \mid E_k(m)=c \text{ 其中 } m \text{ 为有意义之明文}\}$$

则 $s_n(c)=\#\mathbb{K}(c)-1$ 为产生密文 c 的假密钥总数。而所有 n 个字元的假密钥总数的期望值为

$$\begin{aligned}\overline{s_n} &= \sum_{c\in\mathbb{C}^n} p(C=c)s_n(c)\\ &= \sum_{c\in\mathbb{C}^n} p(C=c)(\#\mathbb{K}(c)-1)\\ &= \sum_{c\in\mathbb{C}^n} p(C=c)\#\mathbb{K}(c) - \sum_{c\in\mathbb{C}^n} p(C=c)\\ &= \left(\sum_{c\in\mathbb{C}^n} p(C=c)\#\mathbb{K}(c)\right)-1\end{aligned}$$

其中 n 为很大的整数，且明文空间元素总数等于密文空间元素总数，即 $|\mathbb{M}|=|\mathbb{C}|$。计算

$$\begin{aligned}\log_2(\overline{s}_n+1)&=\log_2\sum_{c\in\mathbb{C}^n}p(C=c)\#\mathbb{K}(c)\\&\geqslant\sum_{c\in\mathbb{C}^n}p(C=c)\log_2(\#\mathbb{K}(c))\\&\geqslant\sum_{c\in\mathbb{C}^n}p(C=c)H(K\mid c)\\&=H(K\mid C^n)\\&=H(K)+H(M^n)-H(C^n)\\&\approx H(K)+nH_L-H(C^n)\\&=H(K)-H(C^n)+n(1-R)\log_2|\mathbf{M}|\\&\geqslant H(K)-n\log_2|\mathbb{C}|+n(1-R)\log_2|\mathbf{M}|\\&(\because H(C^n)\leqslant n\log_2|\mathbb{C}|)\\&=H(K)-nR\log_2|\mathbf{M}|\quad(\text{因为 }|\mathbb{C}|=|\mathbf{M}|)\end{aligned}$$

所以

$$\overline{s}_n\geqslant\frac{|\mathbb{K}|}{|\mathbf{M}^{nR}|}-1,$$

令 $\overline{s}_n=0$，得

$$n_0\approx\frac{\log_2|\mathbb{K}|}{R\log_2|\mathbf{M}|}。$$

从该估算式来看，若自然语言为明文，因其重复率颇高，建议在加密前，先对明文压缩，如此可降低重复率；若压缩算达到最佳化，即 $R\to0$，则 $n_0\to\infty$，这样 Eve 就无法单从密文攻击中破译任何密文。然而，在当代密码技术中，的确是无法完全避免重复率的。甚至以公开密钥密码技术而言，任何人都可取得公开密钥加密编码，要防止的是选择性明文攻击以及选择性密文攻击。但无论如何，还是建议在加密前先压缩，或以其他的方式提高明文的熵值，从而降低明文的重复率。以下，我们借助探讨古典密码的例子，了解如何从估计 Unicity 距离值观察其密码系统的安全度：

例 4.5 假设 Alice 分别以古典密码恺撒挪移法、仿射密码法以及单套字母替换法加密明文，（其中明文为英文，且未经过任何前置处理，或刻意避免任何语言习惯）传讯给 Bob，而 Eve 截收到所有密文。

- （恺撒挪移）此时可能的密钥总数 $|\mathbb{K}|=26$，（含未加密）故 Unicity 距离为

$$n_0\approx\frac{\log_2 26}{0.75\log_2 26}\approx1.33,$$

所以 Eve 只需要截收到字母长度 $\geqslant2$ 的密文，理论上就应该能破译。（其实用暴力攻击即可）

- （仿射密码）此时可能密钥总数 $|\mathbb{K}|=12\times26=312$，（含未加密）故

$$n_0\approx\frac{\log_2 312}{0.75\log_2 26}\approx2.35,$$

所以只需要截收到字母长度 ≥3 的密文，在理想状态下，是可能破译的。

- （单套字母替换）此时可能密钥总数（含未加密）$|\mathbb{K}|=26!$，故

$$n_0 \approx \frac{\log_2 26!}{0.75\log_2 26} \approx 25.1,$$

所以截收到字母长度$\geqslant 26$的密文，理论上就有可能破译；实际上利用频率分析破译，一般英文密文需要截收到 100 个以上的密文字母，才比较可能破译。

这些古典密码法，都非常薄弱。

例 4.6　若 Alice 以单次密码本加密法将长度为 n 的二元字串信息加密，传讯给 Bob。其中可能密钥总数（含未加密）为$|\mathbb{K}|=2^n$，Eve 计算

$$n_0 \approx \frac{\log_2 2^n}{0.75\log_2 2} \approx 1.33n > n,$$

发现这如何也不可能破译，因 Unicity 距离是大于信息的长度。

例 4.7　若 Alice 以 Vigenère 法加密信息长度为 n 的字母给 Bob，而其区块长度为$d>1$，其密钥总数（含未加密）为$|\mathbb{K}|=26^d$，此时

$$n_0 \approx \frac{\log_2 26^d}{0.75\log_2 26} \approx 1.33d,$$

故只需要区块的长度 d 接近信息的长 n，其 Vigenère 密文也是无法破译的。

注意：上述 Unicity 距离只是说明至少要多少密文才会惟一决定密钥值，并未提供任何破译密码的方法。若拿来对 DES 套用，发现其 Unicity 距离比单套字母替换法还短！

第 5 章　对称密钥密码系统

在本章中，我们将介绍现在仍使用的对称密钥密码系统（Symmetric Key Cryptosystems），如 DES、AES 及 IDEA。

在以 DES 为对称密钥密码主流的年代，一般而言，对称密钥密码系统的保密性不如公开密钥密码系统，但由于他们的计算速度很快，在加密大量数据方面，在没有好的替代品出现之前，仍有其必要性。如今应用密码技术的产品，往往是将对称密钥密码与公开密钥密码整合在一起；公开密钥密码的主要用途，还是用来加密对称密钥密码的密钥用来传送密钥以及处理电子签名，而主要内文的加密大都还是用对称密钥密码。

5.1　DES 与 Feistel 密码

美国国家标准局（National Bureau of Standard，即 National Institute of Standards and Technology 的前身）于 1973 年公开招募密码算法，以制定美国国家标准。由 IBM 研发的一种叫做 Lucifer 的对称密钥密码系统，经美国国家标准局交给美国国家安全局（National Security Agency，NSA）修订，成为数据加密标准 DES（Data Encryption Standard），广泛使用了数十年。近年来虽然其安全性已受到置疑，却仍是普遍使用的密码系统，如中国台湾地区的保健 IC 卡，其中加密系统所用的对称密钥密码就是用 Triple DES（Triple DES 为 DES 加密三次，适应目前信息安全需求）；DES 密码符合 20 世纪 70 年代早期电脑硬件配置，为当时工业技术的极品，主要用到简易运算，如二元计算 XOR、循环移位（Cyclic Shift）及置换等，这些将在以下定义中详述。

定义 5.1　令集合 $\mathbb{F}_2 := \{0,1\}$，运算 XOR 定义为

$$a \oplus b := a + b \mod 2\text{（即}0\oplus 0 = 1\oplus 1 = 0, 1\oplus 0 = 0\oplus 1 = 1\text{）。}$$

令 $\mathbb{F}_2^n$ 为所有长度为 n 的 $\mathbb{F}_2$ 字串组成的集合，令字串

$$A = a_1a_2a_3\cdots a_k\cdots a_n \in \mathbb{F}_2^n,$$

若 i、j 为非负整数，运算元循环移位左旋 i 位<<<i 以及右旋j位>>> j 定义为

$$a_k >>> j := a_{j+k} \pmod n, \quad a_k <<< i := a_{k-i} \pmod n\text{。}$$

置换函数 $f: \mathbb{F}_2^n \to \mathbb{F}_2^m$ 即将字串 $A = a_1a_2a_3\cdots a_k\cdots a_n$ 送至字串 $B = b_1b_2b_3\cdots b_h\cdots b_m$ 的函数，其中

$$b_h = a_{l(h)},\quad \text{而 } l: \{1,2,\cdots,m\} \to \{1,2,\cdots,n\},$$

习惯上，函数 l 的值 l(1)、l(1)、…、l(m) 都是用表格形式列出，并以此表示函数 f 。

在密码学的发展历史中，DES 是第一个由研发者主动公开其算法的密码系统，其主要架构源自一种 Feistel 密码的想法，Feistel 密码定义如下。

定义 5.2（Feistel 密码）　Feistel 密码为长度为 2t 位的 $\mathbb{F}_2$ 区块密码（Block Cipher）。令长度为 2t 位的明文为 (L_0, R_0)，其中 L_0 为左半 t 位，R_0 为右半 t 位。加密过程为 r 个回合函数运算，如图 5-1 所示，即

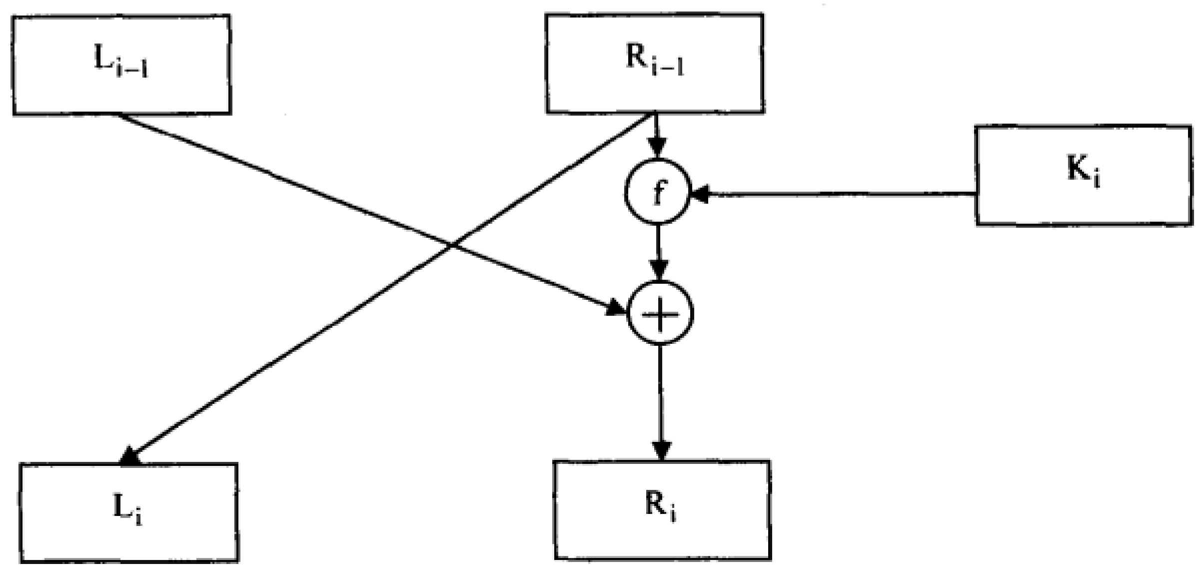


图 5-1　Feistel 密码加密一回合

$$(L_{i-1},R_{i-1})\xrightarrow{K_i}(L_i,R_i)$$

其中 $L_i=R_{i-1}$，$R_i=L_{i-1}\oplus f(K_i,R_{i-1})$，$K_i$ 为密钥 K 所导出的子钥。加密函数为

$$E_K(L_0,R_0)=(L_r,R_r)。$$

DES 密码系统是输入输出区块均为 64 位的 Feistel 密码，其密钥长度为 64 位，而其中有 8 位（即第 8、16、24、…、64 位）为奇偶校验（Parity Bit），所以密钥有效长度为 56 位。而奇偶校验位的设定是为侦测信息传输中所产生的错误。有效的 DES 密钥定义如下。

定义 5.3（DES 密钥）　令字串 $K=k_1k_2k_3\cdots k_{64}\in\mathbb{F}_2^{64}$，则 K 为 DES 密钥 $\Leftrightarrow$

$$\bigoplus_{j=1}^{8}k_{8i+j}=\sum_{j=1}^{8}k_{8i+j}\equiv 1\ (\text{mod } 2)\qquad (0\leqslant i\leqslant 7)。$$

而DES密钥总数为 $2^{56}=72057594037927936$。

例 5.1　一个有效 DES 密钥的 16 进位表达式为

34　13　F1　DF　57　9B　BC　45，

举例而言，其中前8位

$$0x34=(00110010)_2$$

的2进位表达式有奇数个1，其他如0×13、…、0×45的二进位表达式均有奇数个1。

DES 的算法如下：

算法 5.1（DES 加密）　输入≫明文 64 位 $M=m_1m_2\cdots m_{64}$，密钥 64 位 $K=k_1k_2\cdots k_{64}$（含 8 个奇偶校验位）

输出≪密文 64 位 $C=c_1c_2\cdots c_{64}$

（1）（子钥产生）计算 16 个 48 位长的子钥 $K_i\ (i=1,\cdots,16)$，详见算法 5.3。

（2）（初始置换，Initial Permutation，IP）

$$(L_0,P_0)=IP(M),$$

$$L_0=m_{58}m_{50}\cdots m_8,$$

$$R_0=m_{57}m_{49}\cdots m_7,$$

如表 5-1 所示。

表 5-1　初始置换 IP 与终结置换 IP^{-1}

IP								IP^{-1}							
58	50	42	34	26	18	10	2	40	8	48	16	56	24	64	32
60	52	44	36	28	20	12	4	39	7	47	15	55	23	63	31
62	54	46	38	30	22	14	6	38	6	46	14	54	22	62	30
64	56	48	40	32	24	16	8	37	5	45	13	53	21	61	29
57	49	41	33	25	17	9	1	36	4	44	12	52	20	60	28
59	51	45	35	27	19	11	3	35	3	43	11	51	19	59	27
61	53	45	37	29	21	13	5	34	2	42	10	50	18	58	26
63	55	47	39	31	23	15	7	33	1	41	9	49	17	57	25

（3）Feistel 加密 16 回合。

$$L_i = R_{i-1},$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

$$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i)), \qquad (i = 1, \cdots, 16)。$$

如图 5-2 所示。

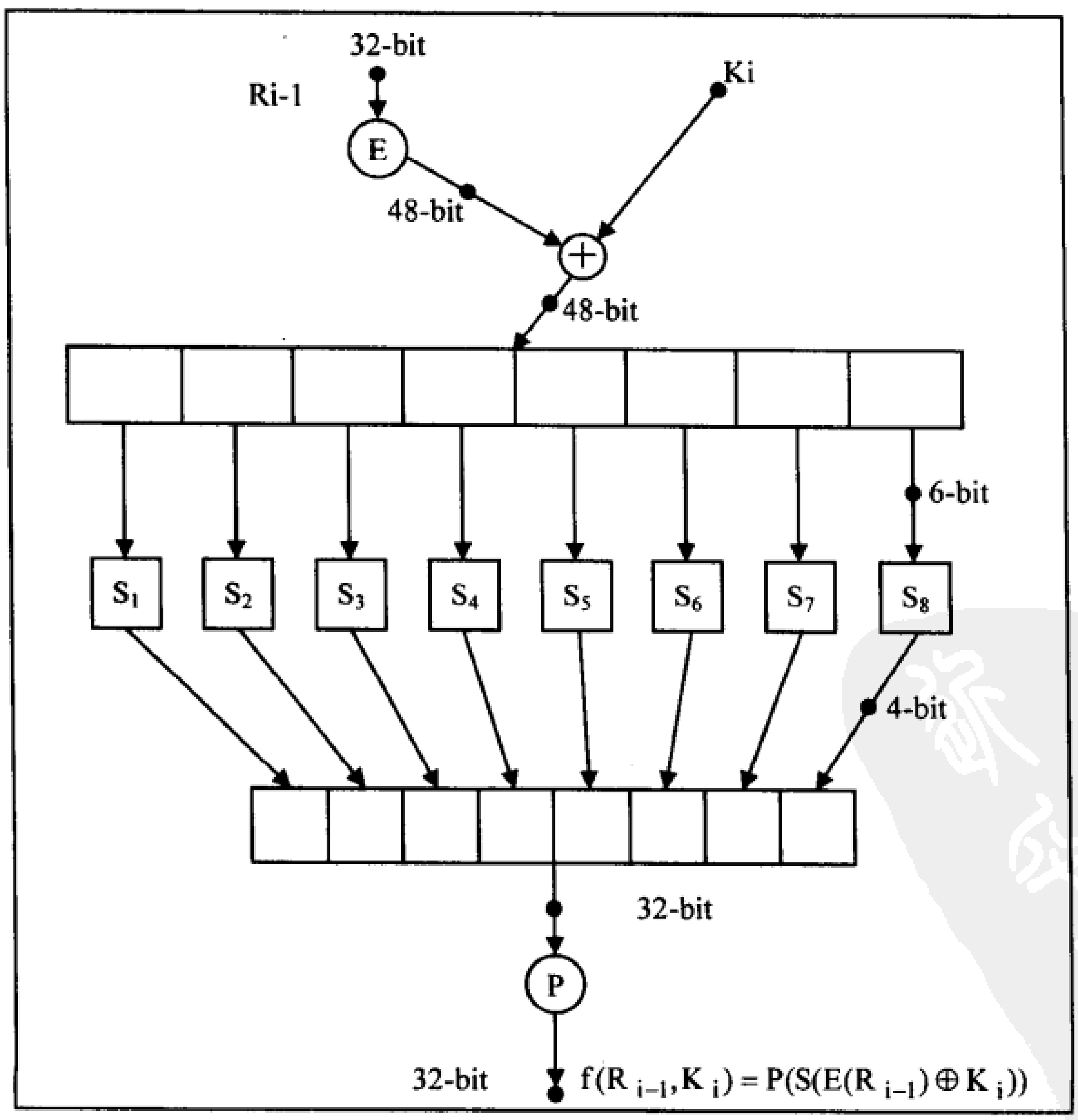


图 5-2　DES f 函数

1）其中函数 $E:\mathbb{F}_2^{32}\to\mathbb{F}_2^{48}$ 定义为

$$T=E(r_1r_2\cdots r_{32})=r_{32}r_1r_2\cdots r_{32}r_1$$

详见表 5-2。

表 5-2 函数 E 与函数 P

E						P			
32	1	2	3	4	5	16	7	20	21
4	5	6	7	8	9	29	12	28	17
8	9	10	11	12	13	1	15	23	26
12	13	14	15	16	17	5	18	31	10
16	17	18	19	20	21	2	8	24	14
20	21	22	23	24	25	32	27	3	9
24	25	26	27	28	29	19	13	30	6
28	29	30	31	32	1	22	11	4	25

2）$B=E(R)\oplus K=B_1B_2B_3\cdots B_7B_8$，其中 B_i 为 6 位长的 $\mathbb{F}_2$ 字串。

3）$U=(S_1(B_1),S_2(B_2),\cdots,S_8(B_8))$，其中

$$S_i:\mathbb{F}_2^6\to\mathbb{F}_2^4$$

即为俗称的 S-Box，详见算法 5.2。

4）$V=P(U)$ 将 $U=u_1u_2\cdots u_{32}$ 送至 $u_{16}u_{17}\cdots u_{25}$，详见表 5-2。

（4）最后一回合对应至 $(R_{16},L_{16})=b_1b_2\cdots b_{64}$。

（5）$C=IP^{-1}(b_1b_2\cdots b_{64})$，其中 IP^{-1} 即为 IP 的反函数。详见表 5-1。

DES 加密流程如图 5-3 所示。

算法 5.2（DES 的 S-Box） 共有 8 个 S-Box，皆可视为系数为 $\mathbb{Z}/16$ 的 4×16 矩阵

$$S[1]、S[2]、\cdots、S[8]\in M_{4\times16}(\mathbb{Z}/16)\text{（详见表5-3）}$$

函数

$$S_i:\mathbb{F}_2^6\to\mathbb{F}_2^4=\mathbb{Z}/16\quad(i=1,2,\cdots,8)$$

$$b_1b_2b_3b_4b_5b_6\mapsto a_{rs},$$

其中 $b_1,b_2,\cdots,b_6\in\mathbb{F}_2$，而且

$$r=2b_1+b_6+1$$
$$s=8b_2+4b_3+2b_4+b_5+1,$$

a_{rs} 为矩阵 S[i] 中的第 (r,s) 项。

例 5.2 令

$$B=b_1b_2b_3b_4b_5b_6=100110\in\mathbb{F}_2^6,$$

经由S-Box S[2] 作用，将值送至 S[2] 的第 (3,4) 项，

（因为 $2+b_6+b_1+1=3$，$b_8b_2+4b_3+2b_4+b_5+1=4$）

即 $11=(1011)_2$。

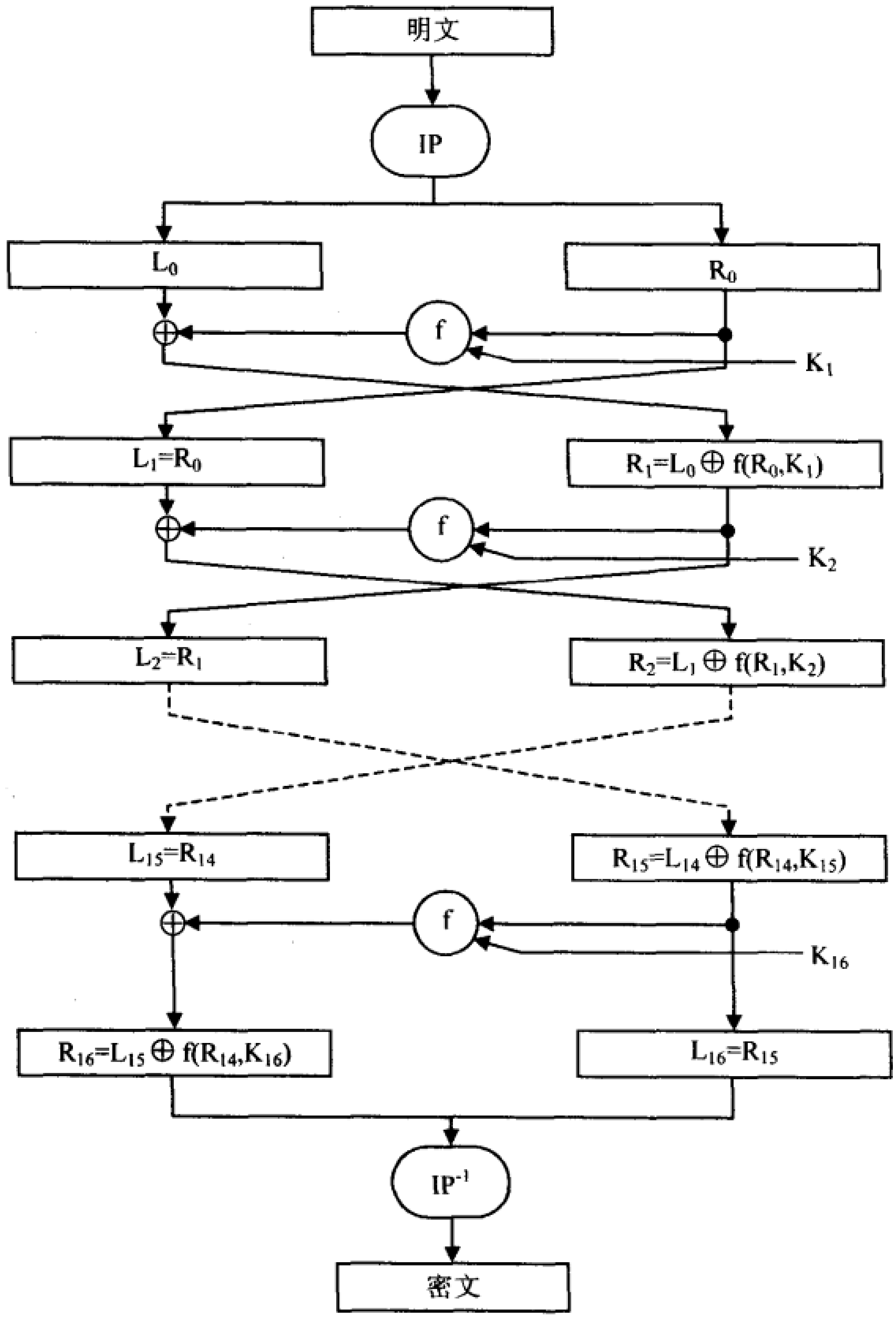


图 5-3　DES 加密流程

表 5-3　DES 的 S-Box

S[1] =	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S[2] =	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

续表

S[3] =	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S[4] =	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S[5] =	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S[6] =	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S[7] =	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S[8] =	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	2	10	8	13	15	12	9	0	3	5	6	11

S-Box 为 DES 算法的核心，本身为非线性函数，它提供 DES 密码安全性的保证，其中：

- S-Box 输入为 6 位，输出为 4 位，在 1970 年代这已是一个芯片最大的容量，这已是当年技术的极限。
- S-Box 所对应的函数为非线性函数，与古典密码大多采用线性函数有相当的差别，这是 DES 密码安全性保证的关键；早年曾有人怀疑，在 S-Box 设计中是否会藏有陷门，这个疑虑也在 IBM 公布 S-Box 的设计内容时消失；另外在加密的使用上，其实也可使用不同的 S-Box。
- 原来 IBM 在设计 S-Box 之际，就已预先防止某些特定攻击法。如在 1990 年由以色列人 Biham 和 Shamir 所发表的一种选择明文攻击，即差分攻击法（Differential Cryptanalysis），早在 1970 年就被 IBM 与 NSA 研发人员所熟知，这项技术，他们保密了近 20 年之久，直至 Biham 和 Shamir 另外独立发表，才真相大白。注[1]

注[1] 有关差分攻击法的描述，读者可参阅 Douglas R.Stinson 所著的“Cryptography Theory and Practice.” 2^{nd} eduition Chapman & Hall/CRC(200)一书的“Differential Cryptanalysis”一节。

算法 5.3（DES 子钥产生） 输入≫64 位密钥 $K = k_1k_2\cdots k_{64}$，含 8 个奇偶校验位。

输出≪16 个 48 位的子钥 K_i $(1 \leqslant i \leqslant 16)$。

（1）定义 $v[i]$ $(i = 1,2,\cdots,16)$ 如下：

$$V[i] = \begin{cases} 1; & i \in \{1,2,9,16\} \\ -1; & i \notin \{1,2,9,16\} \end{cases}。$$

（2）作用函数

$$PC1 : \mathbb{F}_2^{64} \to \mathbb{F}_2^{28} \times \mathbb{F}_2^{28}$$

得 $(C[0], D[0])=PC1(K)$，

其中 $C[0]$ 为左半 28 位，$D[0]$ 为右半 28 位，

$$C[0] = k_{57}k_{49}\cdots k_{36}，\quad D[0] = k_{36}k_{55}\cdots k_4，$$

详见表 5-4。

表 5-4 函数 PC1 与函数 PC2

PC1							PC2					
57	49	41	33	25	17	9	14	17	11	24	1	5
1	58	50	42	34	26	18	3	28	15	6	21	10
19	11	3	60	52	44	36	16	7	27	20	13	2
63	55	47	39	31	23	15	41	52	31	37	47	55
7	62	54	46	38	30	22	30	40	51	45	33	48
14	6	61	53	45	37	29	44	49	39	56	34	53
21	13	5	28	20	12	4	46	42	50	36	29	32

（3）计算 K_i 如下：

```
for(i=1;i<=16;i++)
{
    C[i]=C[i-1]<<<v[i];
    D[i]=D[i-1]<<<v[i];
    K[i]=PC2(C[i],D[i]);
};
```

其中 $PC2(b_1b_2\cdots b_{56}) = b_{14}b_{17}\cdots b_{32} \in \mathbb{F}_2^{48}$ 详见表 5-4。

而 DES 解密的程序即加密的反运算，使用与加密时同一把密钥 K，而子钥 K_i 使用的顺序为倒序，即 K_{16}、K_{15}、…、K_1。其中初始置换 IP 与终结置换 IP^{-1}，因互为反函数，故两作用可互相抵消，即 $(R_{16},L_{16}) = IP(IP^{-1}(R_{16},L_{16})) = IP(C)$，而当中 Feistel 编码解密过程为

$$\begin{aligned} R_{i-1} &= L_i, \\ L_{i-1} &= (L_{i-1} \oplus f(R_{i-1}, K_i)) \oplus f(R_{i-1}, K_i) \\ &= R_i \oplus f(R_{i-1}, K_i)，\quad (i = 16,\cdots,1) \end{aligned}$$

最后再作用 IP^{-1} 即可得明文

$$M = IP^{-1}(IP(M)) = IP^{-1}(L_0, R_0)。$$

5.2　Triple DES 挑战 DES

DES 的前身 Lucifer 原为 112 位密钥，是美国国家标准局将其修订为 56 位的密钥；诚然，密钥的长度减半，使得暴力攻击成为可能。为了适应必要的密码安全要求，只好将已“标准化”的 DES，串联使用多次加密，以求应有的密码安全性，如此一来，加密的效率当然降低，但这在研发出新的替代密码系统前，也是不得不用如此应急手段。然而在使用多次加密时，必须先了解所使用的密码系统是否有特定的代数结构特性，特别是相同密码系统，不同密钥的加密函数是否满足合成运算的封闭性，即

对任意的DES密钥 k_1、k_2，是否存在某密钥 k_3，使得

$$E_{k_3} = E_{k_1} \circ E_{k_2}。$$

如果成立，那么该密码系统就对合成运算“∘”形成群。注[2]如果该密码系统是群，如古典密码是恺撒挪移、区块长度相同的 Vigenère 密码、以及单次密码本，多次加密是没有意义的，因为这样的密码系统多次加密的效果与单次加密的效果是一样的，而且运算时间加倍。幸运的是 DES 并不是群。

定理 5.1　DES 不是群。即存在密钥 k_1、k_2，使得

$$E_{k_3} \neq E_{k_1} \circ E_{k_2}，$$

其中 k_3 为任意DES密钥。

注 5-1　对于这项技术，曾在 IBM 工作的 Coppersmith 指出 DES 不是群，注[3]也表示 IBM 研发人员早已知道。研发密码技术的工作人员与经营密码系统或密码设备的商家，对于特定的技术，只要没人提出，他们会守口如瓶，毕竟“沉默是金”、“言多必失”。

DES 本身并非是群（Group），即集合 $\{E_K \mid K$ 为 DES 密钥$\}$不满足对合成运算“∘”的封闭性。所以似乎多次加密，如 $E_{K_1} \circ E_{K_2}$ 提供了更大的安全性。在实际中，人们常使用 Triple DES，如图 5-4 所示，即

$$\begin{aligned}
&\text{Triple DES-EEE3:}\ E_{K_3} \circ E_{K_2} \circ E_{K_1}\\
&\text{Triple DES-EEE2:}\ E_{K_1} \circ E_{K_2} \circ E_{K_1}\\
&\text{Triple DES-EEE3:}\ E_{K_1} \circ E_{K_2} \circ E_{K_3}\\
&\qquad (K_1 \neq K_2, K_1 \neq K_3, K_2 \neq K_3)
\end{aligned}$$

不用 Double DES 的原因在于，别有用心者可借中途相遇攻击（Meet-in-the-Middle Attack）将原为 112 位的有效长度降为 56 位。

例 5.3（中途相遇攻击，Meet-in-the-Middle Attack）　Alice 与 Bob 使用某不具群结构的对称密钥密码系统如 DES 加密解密传输信息。Alice 使用双次加密 $E_{K_1} \circ E_{K_2}$ 传递给 Bob。假设 K_i 有 n 种可能选取。其中数次，Eve 截收到原文 m_l $(l = 1, 2, \cdots, k)$ 及其密文

注[2] 即满足群的四大公理：封闭性、结合律、存在单位素以及存在反元素。

注[3] 真正完整的证明，还是由 Campbell 与 Wiener 提出，可参阅 K.W.Campbell and M.T.Wiener,“DES IS Not a Group,” Advances in Cryptology-CRYPTO’92 Proceedings,Spring-Verlag,P.512-520。

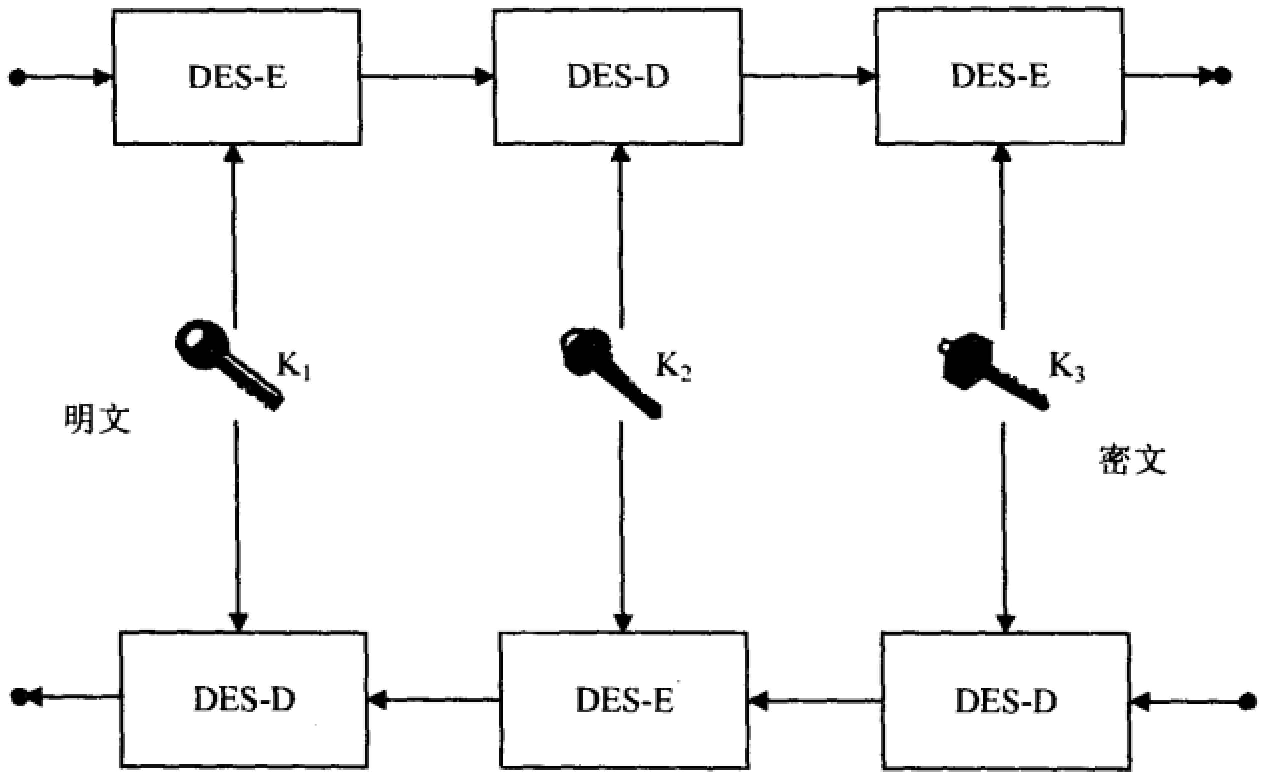


图 5-4 Triple DES-EDE3（用三把不同密钥）

$$c_l = E_{K_1} \circ E_{K_2}(m_l) \quad (l = 1, 2, \cdots, k)$$

并要求其密钥 K_1、K_2。她计算了所有的 $D_K(c_l)$ 及 $E_K(m_l)$ 的值（K 为任意密钥），花了 2n 次计算；比较两者，若恰有一对相同，假设 $D_{K_1}(c_l) = E_{K_2}(m_l)$，则

$$c_l = E_{K_1} \circ D_{K_1}(c_l) = E_{K_1} \circ E_{K_2}(m_l),$$

即得解；若多对相同（应不至于太多对），再与对照文比对计算过滤

$$D_{K_1}(c_l) = E_{K_2}(m_l) \quad (l = 1, 2, \cdots, k),$$

过滤不成立者，直至仅剩一对。（若截收到的原文、密文够多能破解密密钥 K_1、K_2 的概率趋近1）。

用此攻击法，并非像原先用暴力攻击手段需要进行 n^2 次计算才能破解，而是 2n 次计算再加上比较即可，这也就是为什么不采取 Double DES 的原因。而类似的攻击法，也可适用于三次加密，这样可将原先暴力破解所需 n^3 的计算，降至与 n^2 同数量级的计算，就应该可以破解，其安全性长度等同于有效长度为 112 位的密钥。

企图以网络并行分散计算的方式破解 DES 就成了网络热门活动，RSA 公司（RSA security）注[4]在其首页公布悬赏 1 万美金的 DES Challenge III 的挑战，如表 5-5 所示，没多久，他们所公布的密文就被名叫“Deep Crack”的团体破解，所花费时间不过 22 小时 15 分，明文为：

表 5-5 DES Challenge III

```
Ciphertext:
bd 0d de 91 99 60 b8 8a 47 9c b1 5c 23 7b 81 18 99 05
45 bc de 82 01 ab 53 4d 6f 1c b4 30 63 3c ee cd 96 2e
07 c6 e6 95 99 9c 96 46 5a 95 70 02 02 70 98 bd 41 c2
88 a9 f0 2f 8b e5 48 20 d2 a8 a0 6b bf 93 de 89 f6 e2
 52 fd 8a 25 eb d0 7d 96 83 ee a4 2d c8 8d 1b 71
```

注[4] 参阅 http：//www.rsasecurity.com/rsalabs/challenges/des3/index.html。

```
See you in Rome (second AES Conference, March 22-23, 1999)
```

而密钥为：

```
da 4b be f1 6b 6e 98 3d
```

新一代的快速加密算法的研发，显得越生迫切与必要！

5.3 AES

美国国家标准技术局（NIST）于 1997 年公开招标新的密码系统，用以取代早已受人置疑的 DES。在众多的竞争者中，共 15 项不同的加密算法，1998 年入选的有 MARS（IBM 研发）、RC6（RSA Lab 研发）、Rijndael（Joan Daemen 与 Vincent Rijmen 研发）、Serpent（Ross Anderson、Eli Biham 与 Vincent 研发）、Twofish（Bruce Schneier、John Kelsey、Doug Whiting、David Wagner、Chris Hall 与 Niels Ferguson 研发），最后于 2000 年 10 月时才决定由来自比利时的 Joan Daemen 与 Vincent Rijmen 研发的 Rijndael 取得新一代的密码加密标准地位，即 AES（Advanced Encryption Standard）；而 Rijndael（应念作[rain'dal]）是有别于 DES 的，并非前述的 Feistel 密码算法，在区块大小的选择上可用 128 位、192 位以及 256 位三种，计算上也采用多次迭代加密，分别为 10 回合、12 回合以及 14 回合。[注5]

Rijndael 的设计原理与 DES 有很大的不同，在计算的机制上，是以 Galois 体 $\mathbb{F}_{2^8}$ 的算术为计算的基础，借助强韧简洁的代数运算构建成难以破解却能快速加密解密的算法。以 AES-128 为例，即区块为 128 位的 Rijndael，其处理输入方式如下：

定义 5.4（AES-128 的基本单位） AES-128 的区块大小为 128 位，可分为 16 个字节，而每字节含有 8 位，即状态矩阵（State Matrix）$S=(s_{00},s_{10},s_{20},s_{30},s_{01},s_{11},s_{21},s_{31},\cdots,s_{33})$，配置成系数为 $\mathbb{F}_{2^8}$ 的 4×4 矩阵

$$S=\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix};$$

其中每字节 s_{ij} 表为 $\mathbb{F}_{2^8}$ 的元素，Rijndael的 $\mathbb{F}_{2^8}$ 采用模型

$$\mathbb{F}_{2^8} \cong \mathbb{F}_2[t]/(t^8+t^4+t^3+t+1),$$

当中 $t^8+t^4+t^3+t+1$ 为次数 8 不可约的 $\mathbb{F}_2$ 多项式，矩阵中的行向量（Column Vector），

$$\begin{bmatrix} s_{i0} \\ s_{i1} \\ s_{i2} \\ s_{i3} \end{bmatrix}$$

对应于以 $\mathbb{F}_{2^8}$ 为系数的多项式，

注[5] 可参阅美国国家标准局 Rijndael 的官方网站：http://csrc.nist.gov/crypto/toolkit/rijndael。

$$s_{i0} + s_{i1}X + s_{i2}X^2 + s_{i3}X^3 \in \mathbb{F}_{2^8}[X],$$

其中的算术要做模运算 $(\bmod X^4+1)$。（X^4+1 在 $\mathbb{F}_{2^8}[X]$ 中是可约多项式）。

而每回合的128位子钥亦可视为系数为 $\mathbb{F}_{2^8}$ 的4×4矩阵

$$\begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix}。$$

AES-128 算法主要可分为 4 大运算，即

- SubByte
- ShiftRow
- MixColumn
- AddRoundKey

而 AES-128 的加密解密算法可用 4 大运算描写如下：

算法 5.4（AES-128） AES-128 加密算法为：

```
AddRoundKey(S,K[0]);
for(i=1;i<=9;i++)
{
    SubByte(S);
    ShiftRow(S);
    MixColumn(S);
    AddRoundKey(S,K[i]);
};
SubByte(S);
ShiftRow(S);
AddRoundKey(S,K[10]);
```

而 AES-128 解密算法为：

```
AddRoundKey(S,K[10]);
InverseShiftRow(S);
InverseSubByte(S);
for(i=9;i>=1;i--)
{
    AddRoundKey(S,K[i]);
    InverseMixColumn(S);
    InverseShiftRow(S);
    InverseSubByte(S);
};
AddRoundKey(S,K[0]);
```

在 Rijndael 算法中，使用两种 S-Box，如表 5-6 所示，一种用在加密中，另一种用在解密中，互为反运算，即 SubByte 以及 InverseSubByte 均可用代数方式描述。

表 5-6　AES 的 S-Box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

算法 5.5（SubByte 与 InverseSubByte）　SubByte：Rijndael 状态矩阵中任一字节 $s=[s_7,s_6,\cdots,s_0]$ 均视为

$$\mathbb{F}_{2^8}\cong\mathbb{F}_2[t]/(t^8+t^4+t^3+t+1)$$

的元素，即

$$s=s_7t^7+s_6t^6+\cdots+s_0,$$

计算如下：

（1）若 $s=0$，则 $x=0$，否则在 $\mathbb{F}_2[t]/(t^8+t^4+t^3+t+1)$ 中计算 s 的乘法反元素，即以广义辗转相除法计算得 $x=s^{-1}$，$x=[x_7,x_6,\cdots,x_0]$。

（2）计算矩阵乘法

$$\begin{bmatrix}y_0\\y_1\\y_2\\y_3\\y_4\\y_5\\y_6\\y_7\end{bmatrix}=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}x_0\\x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\end{bmatrix}+\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}$$

$y=[y_7,y_6,\cdots,y_0]$ 为经 SubByte 作用后的值。而 InverseSubByte 为 SubByte 的反函数，即先矩

阵左乘上式 8×8 矩阵的反矩阵，再取乘法反元素。

例 5.4（ASE-Box） 以

$$s = t^7 + t^6 + t^3 + t + 1 \in \mathbb{F}_{2^8} \cong \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t + 1)$$

为例，先化为二进制数，再化为 16 进制数，即

$$s = t^7 + t^6 + t^3 + t + 1 \mapsto (11001011)_2 = 0xCB,$$

由AES-Box，s 对应 (C,B) 项，即

$$y = 0x1F = (00011111)_2 = t^4 + t^3 + t^2 + t + 1。$$

当然也可用代数运算检查：

- 首先已知

$$t^2(t^7 + t^6 + t^3 + t + 1) = (t+1)(t^8 + t^4 + t^3 + t + 1) + 1 \in \mathbb{F}_2[t],$$

故 $s = t^7 + t^6 + t^3 + t + 1$ 的乘法反元素为 $x = t^2$（一般须用多项式广义辗转相除法）。

- 矩阵计算：

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

故

$$y = t^4 + t^3 + t^2 + t + 1 = (00011111)_2 = 0x1F$$

与 AES-Box 吻合。

算法 5.6（ShiftRow） ShiftRow 运算为

$$\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} \mapsto \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{11} & s_{12} & s_{13} & s_{10} \\ s_{22} & s_{23} & s_{20} & s_{21} \\ s_{33} & s_{30} & s_{31} & s_{32} \end{bmatrix}。$$

而 InverseShiftRow 为 ShiftRow 的反运算。

算法 5.7（MixColumn） Rijndael 状态矩阵每一行向量，可视为三次 $\mathbb{F}_{2^8}$ 多项式

$$a(X) = a_0 + a_1X + a_2X^2 + a_3X^3,$$

乘上多项式模 $X^4 + 1$：

$$\begin{aligned} c(X) &= t + X + X^2 + (t+1)X^3 \pmod{X^4 + 1} \\ &= 0x02 + 0x01X + 0x01X^2 + 0x03X^3 \pmod{X^4 + 1}。\end{aligned}$$

此运算等价于

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}。$$

而 InverseMixColumn 为 MixColumn 的反运算，就是左乘上式 4×4 矩阵的反矩阵。

AddRoundKey 运算是将状态矩阵与回合子钥进行 XOR 运算。在 AES-128 算法中，尚须从 128 位密钥 K 产生 11 回合子钥 K[0]、K[1]、…、K[10]。

算法 5.8（AES-128 子钥）　AES-128 密钥 K 为 128 位，可分为 4 个 32 位的区块 M[0]、M[1]、M[2]、M[3]。

```
K[0]=M[0];
K[1]=M[1];
K[2]=M[2];
K[3]=M[3];
for(i=1;i<=10;i++)
{
    temp=K[4*i-1];
    temp=SubByte(temp);
    temp=temp XOR RC[i];
    // RC[i]=t^i mod (t^8+t^4+t^3+t+1)
    K[4*i]=K[4*i-4] XOR temp;
    K[4*i+1]=K[4*i-3] XOR K[4*i];
    K[4*i+2]=K[4*i-2] XOR K[4*i+1];
    K[4*i+3]=K[4*i-1] XOR K[4*i+2];
};
```

Rijndael 在设计上，不采取 Feistel 密码的机制。在 Feistel 密码运算中，每一回合左右两半对调位置，一半的位资料只是改变了位置，但其值不变；但在 Rijndael 算法中，所有的位数据都是一般的处理，并不像 Feistel 密码；在 Rijndael 算法中，只需要两回合即可达到充分扩散（Diffusion）的效果。

其 S-Box 设计直接由清楚简易的代数计算完成，即 SubByte，避免任何可能留下陷门的怀疑；Rijndael 的 S-Box 设计基于函数 $x \mapsto x^{-1}$（在 $\mathbb{F}_{2^8}$）和一仿射效应（Affine Mapping），这样的计算是高度非线性的，对于能有效攻击 DES 的差分分析（Differential Cryptanalysis）或是线性分析（Linear Cryptanalysis），都因此而不能奏效。而 MixColumn 对 Byte 之间产生扩散作用。另外次密钥 K_0、K_1…的产生也用到 S-Box（即 SubByte），使得各子钥之间可能的对称性关系也得以去除。以 AES-128 为例，它是采用 10 回合迭代计算而成，但随着需求变化，也可轻易增加迭代运算的回合数。截至目前为止，还没有能以暴力攻击找出 7 回合以上运算的关联性。

5.4　IDEA

IDEA 即 International Data Encryption Algorithm（国际数据加密算法），是一种对称密钥加密方法，在 1990 年由 Lai 和 Massey 所设计提出。IDEA 是以 128 位的密钥对称区块为 64 位

的明文，经连续 8 次加密后产生 64 位密文的一种加密算法。

IDEA 的设计核心是建立在三种非常简易的代数运算上：

- 以位为单位的 XOR 运算，用 $\oplus$ 表之。
- 在加法群 $\mathbb{Z}/2^{16}$ 上的加法，即模加法以 16 位为单位，用 $\boxplus$ 表之。
- 在乘法群 $(\mathbb{Z}/2^{16}+1)^{\times}$ 上的乘法，即（ $\bmod 2^{16}+1=65537$ ）的模乘法，须知

$$F_4=2^{16}+1$$

费马数（Fermat Number）为质数，故乘法群的元素个数为

$$|(\mathbb{Z}/2^{16}+1)^{\times}|=2^{16}+1-1=2^{16},$$

以 16 位为计算单位，其中以“0”为代表

$$\text{“}-1\equiv 2^{16}\ (\bmod 2^{16}+1)\text{”}$$

用符号 $\odot$ 代表模乘法（其中此运算也可视为 IDEA 的 S-Box）。

基本上，此三种运算彼此之间没有分配律（以 $\mathbb{F}_{2^{16}}$ 代表此 16 位计算单位）：

如

$$\exists a,b,c\in\mathbb{F}_{2^{16}} \text{ 使得}$$

$$a\boxplus(b\odot c)\neq(a\boxplus b)\odot(a\boxplus c)$$

$$a\boxplus(b\oplus c)\neq(a\boxplus b)\oplus(a\boxplus c)$$

注 5-2 费马数 $\mathbb{F}_4=2^{2^4}+1$ 为质数，使得 IDEA 代数运算在 16 位上计算非常简易，但费马数 $\mathbb{F}_5=2^{2^5}+1$ 不为质数，这就无法将 IDEA 的代数运算直接推广至 32 位。

例 5.5 令 $a=0$、$b=1$、$c=2$，则

$$\begin{aligned}a\odot(b\boxplus c)&=0\odot(1\boxplus 2)\\&=0\odot(1+2\ \ (\bmod 2^{26}))\\&=-1\times 3\ \ (\bmod 2^{26}+1)\\&=65537-3=65534\ 。\end{aligned}$$

而

$$\begin{aligned}(a\odot b)\boxplus(a\odot c)&=(0\odot 1)\boxplus(0\odot 2)\\&=(-1\times 1\ \ (\bmod 2^{26}+1))\\&\quad\boxplus(-1\times 2\ \ (\bmod 2^{26}+1))\\&=(2^{26}+1-1)\boxplus(2^{26}+1-2)\\&=2^{26}+2^{26}-1\ \ (\bmod\ 2^{26})\\&=65535\ 。\end{aligned}$$

所以可得

$$a\odot(b\boxplus c)\neq(a\odot b)\boxplus(a\odot c)\ 。$$

IDEA 计算的核心是 MA-Box，这提供了扩散的效果，其运算结构为 4 个 16 位的输入和 2 个 16 位的输出，可描述如下。

算法 5.9（MA-Box） 输入 $\gg$ 4 个 16 位单位 F_1、F_2、K_5、K_6。

输出 $\ll$ 2 个 16 位单位 G_1、G_2。

$$temp = F_1 \odot K_5;$$
$$G_2 = temp \boxplus F_2;$$
$$G_2 = G_2 \odot K_6;$$
$$G_1 = temp \boxplus G_2;$$

可参阅图5-5。

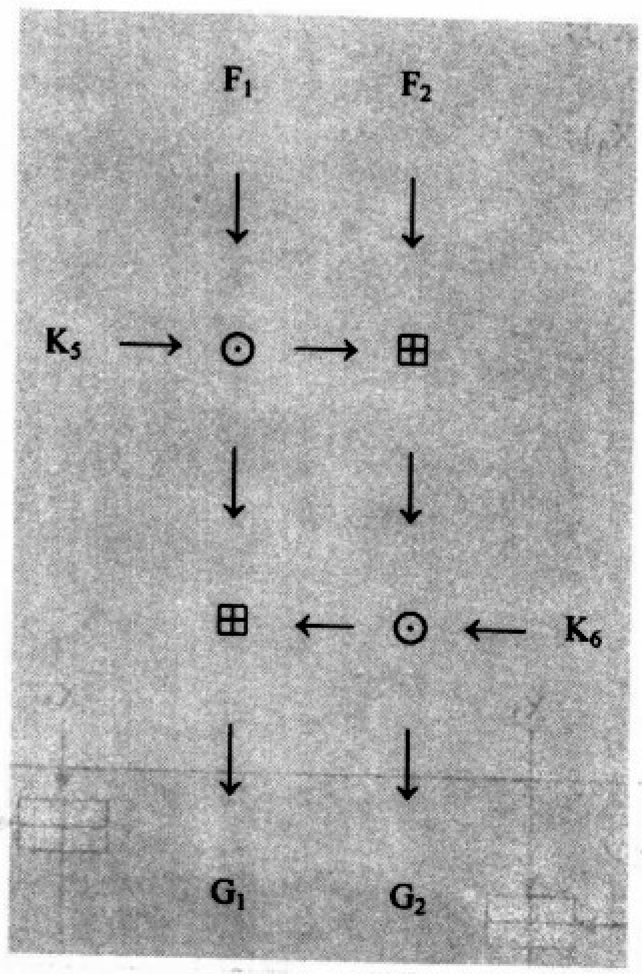


图 5-5　IDEA 的 MA-Box

IDEA 的算法主要分为 8 个回合计算，区块单位为 64 位，而密钥为 128 位，其算法如下。

算法 5.10　输入≫64 位明文 $M = m_1m_2\cdots m_{64}$，128 位密钥 $K = k_1k_2\cdots k_{128}$。

输出≪64 位密文 $C = Y_1 \| Y_2 \| Y_3 \| Y_4$，其中 $Y_i\ (i=1,\cdots,4)$ 为 16 位单位区块。

（1）（子钥产生）产生 52 把子钥，$K_1^{(r)}$、…、$K_6^{(r)}$，（其中 $r=1,\cdots,8$ 为第 r 个回合的子钥）以及 $K_1^{(9)}$、…、$K_4^{(9)}$。

（2）将明文 M 分割成 4 个 16 位区块

$$M = X_1 \| X_2 \| X_3 \| X_4,$$

其中

$$X_1 = m_1m_2\cdots m_{16}$$
$$X_2 = m_{17}m_{18}\cdots m_{32}$$
$$X_3 = m_{33}m_{34}\cdots m_{48}$$
$$X_4 = m_{49}m_{50}\cdots m_{64}\ 。$$

（3）

```
for(r=1;r<9;r++)
{
```

MixKey:

$X_1 = X_1 \odot K_1^{(r)}$;

$X_4 = X_4 \odot K_4^{(r)}$;

$X_2 = X_2 \boxplus K_2^{(r)}$;

$X_3 = X_3 \boxplus K_3^{(r)}$;

MA-Box:

$temp = K_5^{(r)} \odot (X_1 \oplus X_3)$;

$G_2 = temp$;

$temp = K_5^{(r)} \odot (X_1 \oplus X_3)$;

$G_1 = G_2 \odot K_6^{(r)}$;

Renew:

$X_1 = X_1 \oplus G_1$;

$X_4 = X_4 \oplus G_2$;

$temp = X_2 \oplus G_2$;

$X_2 = X_3 \oplus G_1$;

$X_3 = temp$;

}

可参阅图 5-6。

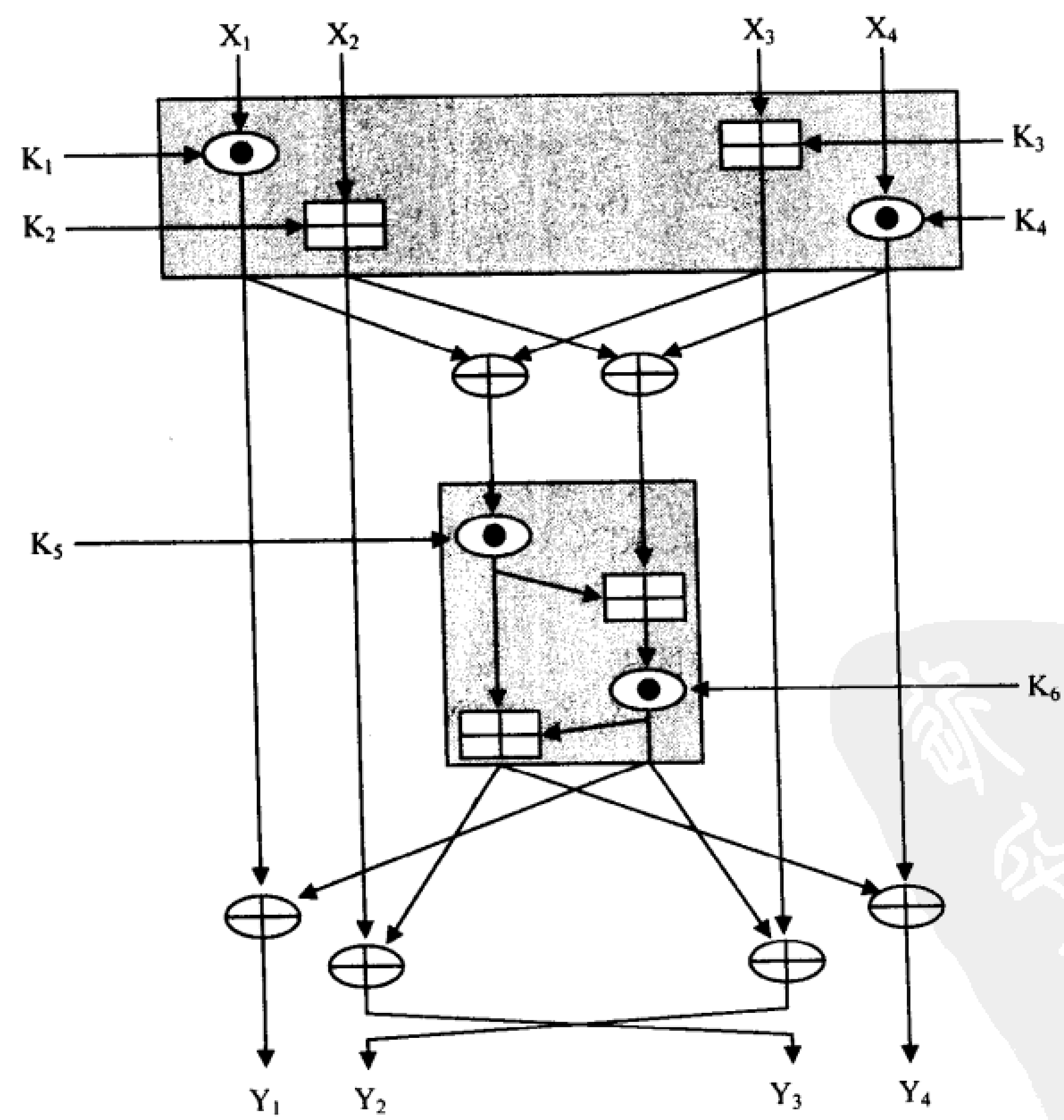


图 5-6 IDEA 的一回合作用

（4）（输出）

$Y_1 = X_1 \odot K_1^{(9)}$；

$Y_4 = X_4 \odot K_4^{(9)}$；

$Y_2 = X_3 \boxplus K_2^{(9)}$；

$Y_3 = X_2 \boxplus K_3^{(9)}$；

算法 5.11（IDEA 子钥产生）　输入≫128 位密钥 $K = k_1k_2 \cdots k_{128}$。

输出≪52 把 16 位的子钥 $K_i^{(r)}$（$i = 1,\cdots,6$，$r = 1,\cdots,8$）以及 $K_j^{(9)}$（$j = 1,\cdots,4$）。

（1）52 把 16 位的子钥的顺序为

$$K_1^{(1)}, L\, K_6^{(1)}, K_1^{(2)}, L, K_6^{(2)}, L\, K_1^{(8)}, L\, K_6^{(8)}, K_1^{(9)}, L, K_4^{(9)}。$$

（2）将 K 分割成 8 个 16 位区块，即前 8 把子钥。

（3）将 K 左旋 25 位，分割成 8 个 16 位区块，即另 8 把子钥；重复此项步骤直至 52 把子钥都生成。

值得一提的是，对 IDEA 的安全性评价颇高，以 Phil Zimmerman 一人之力发展起来的 eMail 加密软件 PGP（Pretty Good Privacy）从第二版起，就加入了 IDEA 算法的加密信息的选择。

5.5　区块密码加密模式

以上所提的密码如 DES、IDEA、AES 都是区块密码，以后介绍的各种公开密钥密码也都是区块密码，区块密码是以一定大小的区块为单位加密，最后一区块也要填补成一个区块数据予以加密。任一特定的加密算法，都可以不同的模式加密，一般而言，可以分为 4 种加密模式，即 ECB（Electronic Code Book，电子编码簿）、CBC（Cipher Block Chaining，密码区块串联）、CFB（Ciphertext Feedback，密文回馈）和 OFB（Output Feedback 回馈）。

电子编码簿模式 ECB，如图 5-7 所示。

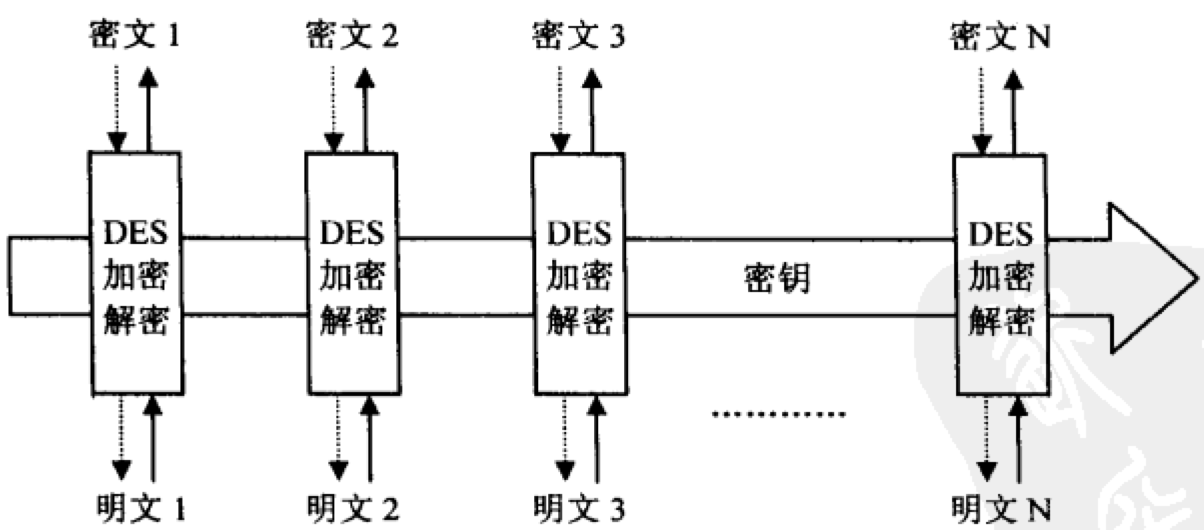


图 5-7　ECB 模式 DES 演算

最自然、最简易的区块密码加密模式，就是将信息 m 分割成等大小的区块密文，即

$$m = m_1 \| m_2 \| \cdots \| m_N，$$

当中最后一个区块填补成一个区块数据，然后分别以固定加密密钥下的加密函数 $E_{K_e}(\cdot)$ 加密，即

$$c_i = E_{K_e}(m_i) \quad (i = 1, 2, \cdots, N),$$

得密文

$$c = c_1 \| c_2 \| \cdots \| c_N。$$

这种加密的模式是 ECB，而解密的方式是将每区块直接解密

$$m_i = D_{K_d}(c_i) \quad (i = 1, 2, \cdots, N),$$

得明文。ECB 的优点是各区块的运算可独立运行，即使在部分区块传输运算上发生错误，也不会影响到其他区块，加密解密都可并行处理运算。其缺点为，如果文件中多次重复的明文又符合区块的大小，可能会产生相同密文，这对于统计式的攻击相当不利，不适合长文件的加密处理，ECB 模式所产生的密文即使被剪贴、替换，也不易被发现。

密码区块串联模式 CBC，如图 5-8 所示。

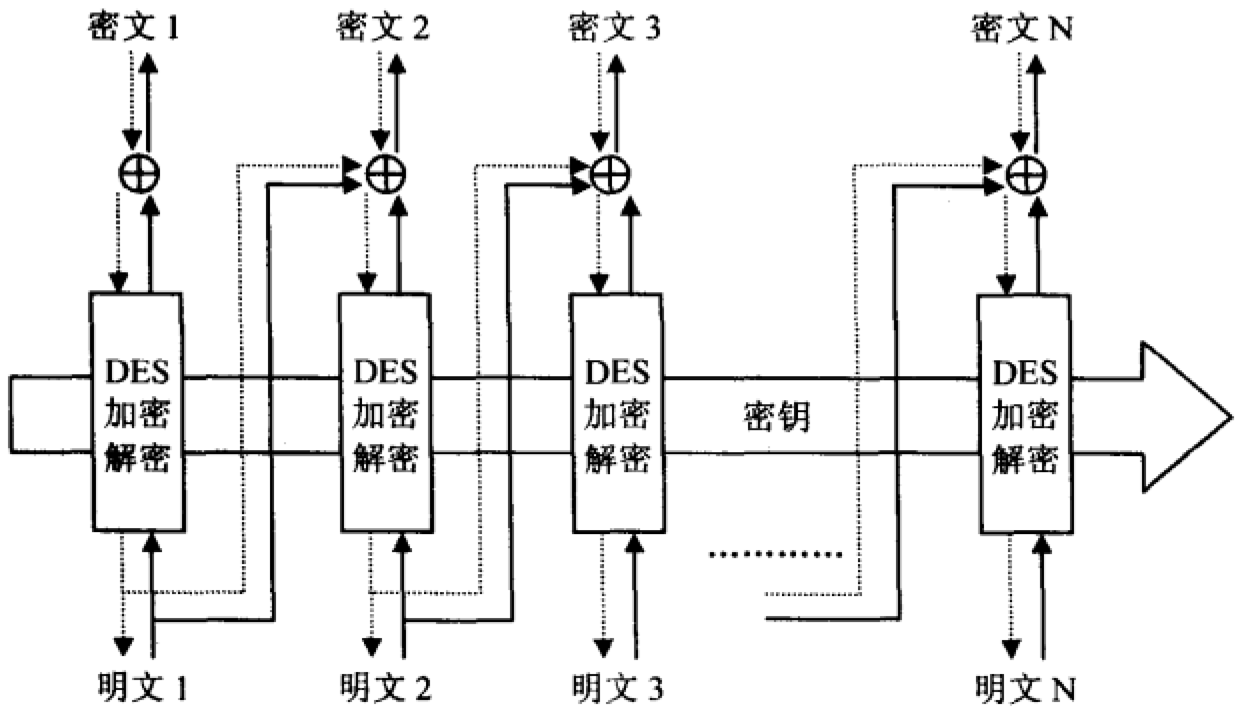


图 5-8 CBC 模式 DES 演算

要避免在 ECB 模式中可能产生的缺点，可以用串联的方法。串联本身就是一种回馈（Feedback）的机制，与前一区块的加密有关。实际的做法如下：

- 先选取双方同意的一个区块大小的数据 IV，即初始向量（Initial Vector），该向量可公开。
- （加密）将信息 m 分割成同等大小的区块明文，即

$$m = m_1 \| m_2 \| \cdots \| m_N,$$

当中最后一个区块 m_N 要补足一个区块单位的数据，然后分别用固定的加密密钥 K_e 的加密函数 $E_{K_e}(\cdot)$ 加密：

$$c_0 = IV, \quad c_i = E_{K_e}(c_{i-1} \oplus m_i) \quad (i = 1, 2, \cdots, N),$$

得密文

$$c = c_1 \| c_2 \| c_3 \| \cdots \| c_N。$$

- （解密）用解密密钥 K_d 下的解密函数解密 D_{K_d}，其中 $D_{K_d}(E_{K_e}(m_i)) = m_i$，计算

$$c_0 = IV, \quad m_i = c_{i-1} \oplus D_{K_d}(c_i) \quad (i = 1, 2, \cdots, N)。$$

一般而言，使用不同的初始向量 IV，对相同的明文，用 CBC 模式加密应该会得到不同密文。但由于加密的计算与前次区块的明文有关，所以如果明文区块的顺序改变或整块密文区块被替换，解密就变成不可能，这样可防止他人的蓄意攻击。

某区块传输错误会影响后一区块的计算：就解明文区块 m_i 而言，密文区块 m_i 以及 m_{i+1} 产生错误，但以下的区块 m_{i+2} 、 m_{i+3} 、…不会收此影响。

另外，与 ECB 不同之处，就是 CBC 无法用并行处理方式加密，而解密时，由于在作 XOR 运算前就已知道 c_i 及 c_{i-1} ，所以 CBC 模式解密时可以并行处理。

密文回馈模式 CFB

上述的 CBC 模式非常适用于大量信息的加密，然而在实时（Real-Time）通信的应用中，如 Bob 收到 Alice 传送的密文后，想立即解密时，就不适用了，此时效率就变成非常重要的问题，这种情况下 CFB 模式加密就可派上用场了。与 ECB、CBC 不同，CFB 模式加密以 8 位为单位（或更小的单位），无须等待整个区块的信息全部到达才进行解密，这在互动式的电脑通信上有实用的价值。但是要注意，CFB 模式加密、解密只用到加密函数 $E_K(\cdot)$ ，未用到解密函数，因此无法适用于公开密钥密码系统，只能适用于对称密钥密码系统。CFB 的运作方式如下：

- 与 CBC 模式一样，固定一个区块大小的初始向量 X_1 。将明文分割成 s 位的小单位
$$m = m_1 \| m_2 \| \cdots,$$
其中每个区块的大小为 n 位且 $s \mid n$ 。
- （加密）当 $j = 1,2,3,\cdots$ 时，计算
$$c_j = m_j \oplus L_s(E_k(X_j))$$
$$X_{j+1} = R_{n-s}(X_j) \| c_j \ ,$$
其中 $L_s(X)$ 是代表 X 的左s位， $R_{n-s}(X)$ 是代表 X 的左(n-s)位。
- （解密）当 $j = 1,2,3,\cdots$ 时，计算
$$m_j = c_j \oplus L_s(E_k(X_j))$$
$$X_{j+1} = R_{n-s}(X_j) \| c_j$$

以 DES 为例，64 位为一个区块单位，8 位为小单位，以 CFB 模式加密，64 位暂存器的初始值为 X_1 。则

$$\begin{aligned}
c_1 &= m_i \oplus L_s(E_k(X_1)) \\
X_2 &= R_{56}(X_1) \| c_1 \\
X_3 &= L_{56}(X_2) \| c_2 = L_{48}(X_1) \| c_1 \| c_2 \\
X_4 &= L_{56}(X_3) \| c_3 = \cdots = L_{40}(X_2) \| c_1 \| c_2 \| c_3 \\
&\vdots \\
X_9 &= \cdots = c_1 \| c_2 \| c_3 \| \cdots \| c_8
\end{aligned}$$

经过 8 次运算，初始值 X_1 才完全消失，并被 c_j 取代，之后暂存器的值为

$$\begin{aligned}
X_{10} &= c_2 \| c_3 \| \cdots \| c_9 \\
X_{11} &= c_3 \| c_4 \| \cdots \| c_{10} \\
X_{12} &= c_4 \| c_5 \| \cdots \| c_{11} \\
&\vdots
\end{aligned}$$

在实际应用中，信息传递的错误对 CFB 模式的影响会随着暂存器值变化的情况影响后面连续 8 个小单位密文的解密。假设密文

$$c = c_1 \| c_2 \| c_3 \| \cdots ,$$

当中 c_1 在信息传递中发生错误，变成 $\hat{c}_1$，暂存器的值变化如下

$$\hat{X}_2 = {*}{*}{*}{*}{*}{*}{*} \| \hat{c}_1$$

$$\hat{X}_3 = {*}{*}{*}{*}{*}{*} \| \hat{c}_1 \| c_2$$

$$\vdots$$

$$\hat{X}_9 = \hat{c}_1 \| c_2 \| c_3 \| c_4 \| \cdots \| c_8$$

$$X_{10} = c_2 \| c_3 \| \cdots \| c_9$$

暂存器值在 X_{10} 之后又恢复正确，因此明文小区块 m_{10}、m_{11}、…又恢复到正确值。

输出回馈模式 OFB

OFB 模式与 CFB 非常类似，与 CFB 模式加密一样不适用于公开密钥密码系统，只能用于对称密钥密码系统，其演算过程如下：

- 与 CFB 模式一样，先固定一个区块大小的初始向量 X_1。将明文分割成 s 位的小单位

  $$m = m_1 \| m_2 \| \cdots$$

  其中每区块的大小为 n 位且 $s | n$。

- （加密）当 $j = 1, 2, 3, \cdots$ 时，计算

  $$c_j = m_j \oplus L_s(E_k(X_j))$$

  $$X_{j+1} = E_k(X_j)$$

- （解密）当 $j = 1, 2, 3, \cdots$ 时，计算

  $$m_j = c_j \oplus L_s(E_k(X_j))$$

  $$X_{j+1} = E_k(X_j)$$

在 CFB 模式中，若有一区块的密文在传输上产生错误，在解密时，只有该区块会解密错误，其他的区块并不会受到影响，其缺点为比 CFB 更易遭受攻击。在实际应用中，比其他模式更适用于在不太稳定的信道上加密，如人造卫星通信中的加密。

第 6 章 RSA 密码

1976 年 Whitefield Diffie 和 Martin Hellman 提出了公开密钥（Public Key）密码的一种加密解密新想法注[1]。稍后，在 1977 年《科学美国人》报道了当时 MIT 的 Ronald Rivest、Adi Shamirt 和 Leonard Adleman 开创的一种“数百万年才能破解”的公开密钥密码系统（Public Key Cryptosystem），注[2]即 RSA 密码系统，并提出资金 100 美元的所谓 RSA-129 挑战。如今 RSA 密码系统已经成为使用最广、安全性高的公开密钥密码系统，RSA 密码、RSA 算法与 RSA 数字签名等，早已被学术界、业界广泛研究与使用。

但是，1997 年英国情报官方资料显示，前身为位于 Bletchley Park 的政府代码暨密码学校（Government Code and Cipher School）、政府通信总部（Government Communication Headquarters）才是最早发明公开密钥密码系统的单位。其间 James Ellis 早在 1970 年已有公开密钥密码的想法，而被网罗才三个星期的剑桥大学毕业生、主攻数论的 Clifford Cocks 在 1973 年就已发明我们现称为 RSA 的算法，Diffie-Hellman 密钥交换也在 1974 年被 Malcom Williamson 发现，公开密钥密码的主要技术都在 1974 年被英国情报单位掌握，足足比学术界早数年之久，英国情报单位在密码学方面的研究实力实在不容小觑！注[3]在本章中，我们讨论如下的内容：

- 公开密钥密码系统
- RSA 算法
- RSA 的数论背景
- RSA 数字签名
- 同时进行 RSA 加密和 RSA 数字签名
- RSA-129 挑战与因数分解
- 二次筛法与 Pollard 的 p-1 法
- 利用 RSA 私钥因数分解
- RSA 密码系统实用的注意事项
- Wiener 低幂次 d 攻击
- Rabin 密码

6.1 公开密钥密码系统

在一个公开密钥密码系统中，任何使用者都可以取得其他使用者的加密密钥（即公开密

注[1] 可参阅 Diffie and Hellman，New directions in cryptography.IEEE Trans.Inform.Theory IT-22 P.644-654，1976。

注[2] 可参阅 Rivest，Shamir and Adlman,A Method for Obtaining Digital Signature and Public-Key Cryptosystems.Communication of the A.C.M,21.No.2p.120-126，1978。

注[3] 这些英国情报单位在密码学的研究成果可参阅：http://www.cesg.gov.uk/publications/index.htm#nsecret。

钥)，假设 Alice 欲通过此公开密钥密码系统加密传信给 Bob，她可以先取得 Bob 的加密密钥，通过加密函数 $E_{Bob}(\cdot)$ 将信息加密成密文传递给 Bob，Bob 在收到 Alice 传来的密文后，便可通过自己的解密密钥（私钥），用解密函数 $D_{Bob}(\cdot)$ 解密。在这些步骤中用到以下性质：

（1）将一已加密的信息 m 解密，即可还原 m，即

$$D_{Bob}(E_{Bob}(m)) = m$$

（2）加密以及解密函数 $E_{Bob}(\cdot)$、$D_{Bob}(\cdot)$ 必须是容易计算的。

（3）公开加密函数 $E_{Bob}(\cdot)$，并不会提供任何简易计算解密函数 $D_{Bob}(\cdot)$ 的方法，在实际应用中，这意味着只有 Bob 才能将任何经加密函数 $E_{Bob}(\cdot)$ 加密的信息有效地解密，也只有 Bob 知道“陷门”（Trapdoor）从而有效计算 $D_{Bob}(\cdot)$。

（4）若将任何信息 m 先用解密函数 $D_{Bob}(\cdot)$ 运算，再用加密函数 $E_{Bob}(\cdot)$ 运算，亦可还原信息 m，即

$$E_{Bob}(D_{Bob}(m)) = m \text{。}$$

一种加密法（解密法）一般而言，包括了算法以及密钥，泄漏了加密算法就意味着公开其加密密钥。

定义 6.1（单向陷门函数，Trapdoor One-Way Function） 函数 $E_{Bob}(\cdot)$ 若满足性质（1）、（2）、（3），就称为单向陷门函数，若性质（1）、（2）、（3）、（4）皆满足，就称为单向陷门置换（Trapdoor One-Way Permutation）。

公开密钥密码系统的实践需要单向陷门函数，虽然 Diffie 及 Hellman 已经提到此概念，但未能提出任何单向陷门函数的实例，单向陷门函数之所以称为“单向”，就是这样的函数可以轻易地计算一个方向，但另一个方向却是非常难以计算的。而满足性质（4）的单向陷门置换，也提供了一种数字签名的方式，在此情形下，加密函数与解密函数互为反函数。而“陷门”意味着，只有加上特定的额外信息，才能将任何经加密函数 $E_{Bob}(\cdot)$ 加密的信息有效地解密；以 RSA 公开密钥密码系统而言，陷门信息就是对模数因数分解的信息。

一般而言，公开密钥密码系统的计算远比传统的对称密钥密码系统缓慢，公开密钥密码系统也可嵌入对称密钥密码系统，即用公开密钥密码系统将双方的对称密钥密码的密钥加密传送，接收方再用公开密钥密码系统的解密函数解开所传送的密钥，然后用此密钥解开密文，因此公开密钥密码系统提供了解决传统对称密钥密码密钥分送问题的方法，通信安全与加密解密速度得以兼顾，这种加密方式称为“数字封装”（Digital Envelope）。试想，在一个加密的场景里：Alice 与 Bob 用两套系统来传递信息，假设一套公开密钥密码系统为 RSA，另一套为对称密钥密码系统 IDEA。在这个场景中，主要是用到三把密钥：

- IDEA 密钥，其所对应的函数作用为 IDEA 加密以及 IDEA 解密；
- RSA 加密密钥，其函数作用为 RSA 加密。
- RSA 解密密钥，其函数作用为 RSA 解密。

主要加密、解密过程要用到 IDEA，但 IDEA 的密钥只有 Alice 知道，Bob 若无 IDEA 密钥，也无法破解。如何传送 IDEA 密钥，而又不怕被怀有恶意的 Eve 夺取？其做法就是将 IDEA 密钥“锁在箱子里”，而这箱子，只有 Bob 的 RSA 解密密钥才能打开。譬如 eMail 加密软件 PGP（Pretty Good Privacy）就是用数字封装加密解密。自第二版起，公开密钥密码系统可用 RSA，对称密钥密码系统可用 IDEA，如图 6-1 所示。

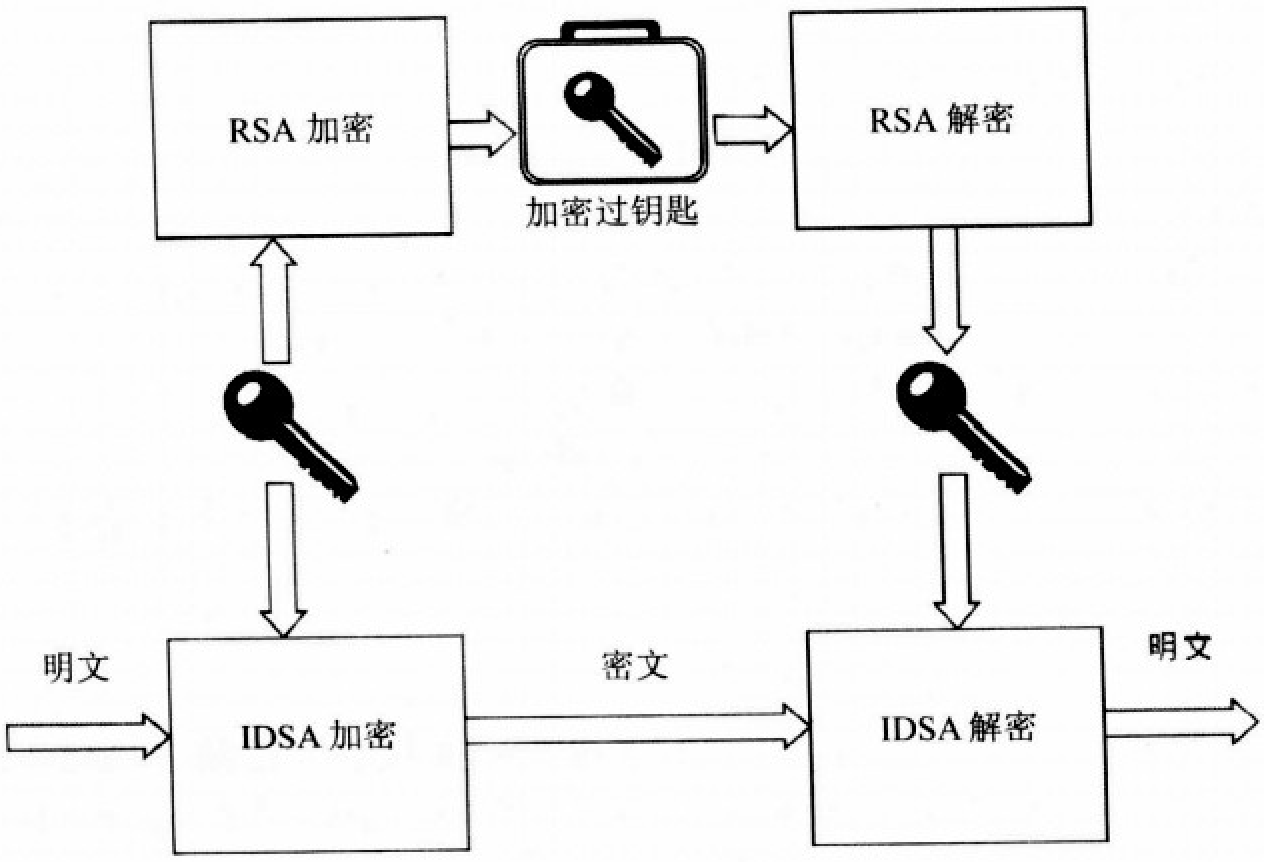


图 6-1　数字封装

6.2　RSA 算法

RSA 算法究竟是什么？假设 Alice 想将信息用 RSA 算法加密传至 Bob，而 Bob 如何将其解密解读？而怀有敌意的 Eve 能否破解它？这要如何操作？我们先将加密解密过程整理如下。

算法 6.1（RSA 算法）　令 Alice 想将明文 m 加密成密文 c 传至 Bob，而 Bob 将密文 c 解密还原成明文 m，如图 6-2 所示。

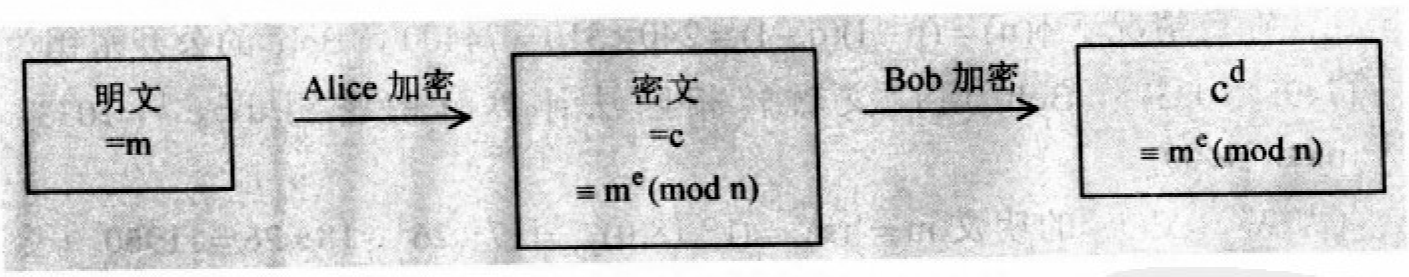


图 6-2　RSA 算法

（1）（密钥产生）Bob 取相异质数 p、q（保密），计算 RSA 模数（RSA Modulus）$n = pq$，将其公开，取 e 为加密密钥将其公开，其中 e 必须与 $\phi(n)$ 互质，在此情况下

$$\phi(n) = (p-1)(q-1),$$

Bob 的公开密钥为 (n,e)；Bob 计算 d 为解密密钥（保密），(n,d) 为 Bob 的私钥（Private Key）其中

$$ed \equiv 1 \pmod{\phi(n)}。$$

（2）（加密）Alice 取得 Bob 的公开密钥 (n,e)，用加密函数计算

$$c = E(m) \equiv m^e \pmod{n},$$

将密文 c 传给 Bob。

（3）（解密）Bob 用解密函数计算

$$m = D(c) \equiv c^d \pmod n$$

解密还原成明文。

注 6.1 RSA 算法为公开密钥密码算法，不同于对称密码算法如 Enigma、DES、AES、IDEA，公开密钥 (n,e) 必须让发送方知道，而私钥 (n,d) 的保密至关重要。其中：

（1）RSA 算法可成立的关键在于，加密密钥 e 与解密密钥 d 满足

$$ed \equiv 1 \pmod{\phi(n)}$$

由此可用 Euler 定理推导出加密函数 $E(\cdot)$ 与解密函数 $D(\cdot)$ 互为反函数，将在下节定理（6.2）中证明。

（2）Euler-Phi 函数的定义为

$$\phi(n) := \#\{a \in \mathbb{N} \mid a < n, \gcd(a,n) = 1\}。$$

若 $\phi(n) = (p-1)(q-1)$ 已知，解密密钥 d 可由广义辗转相除法（Extended Euclidean Algorithm）计算得出，值 $\phi(n)$、p 与 q 千万要保密；若 $\phi(n)$ 已知，则 $p + q = n + 1 - \phi(n)$ 可用一元二次公式解 $x^2 - (n+1-\phi(n))x + n = 0$ 得

$$p、q = \frac{n+1-\phi(n) \pm \sqrt{(n+1-\phi(n))^2 - 4n}}{2}。$$

（3）RSA 加密函数 $E(\cdot)$ 为单向陷门函数，一般相信 RSA 算法的安全性在于整数 n 的质因数分解的困难度，n 值一般建议为 1024 位（309 位十进制整数），对于中长期使用建议为 2048 位，随着数论学家在质因数分解领域的发展以及电脑信息科技的进步，n 值大小的要求还会再提升。

我们先看 RSA 算法的简易例子。

例 6.1

- （密钥产生）Bob 取质数 $p = 241$、$q = 311$，计算 $n = pq = 74951$，取 $e = 1033$ 为加密钥，在此情况下 $\phi(n) = (p-1)(q-1) = 240 \times 310 = 74400$，Bob 的公开密钥为 $(n,e) = (74951,1033)$；Bob 用广义辗转相除法计算解密密钥 $d \equiv e^{-1} = 1033^{-1} \equiv 3097 \pmod{74400}$。
- （加密）Alice 的明文 $m = \text{'rsa'} = (17,18,0)_{26} = 17 \times 26^2 + 18 \times 26 = 11960$（令 a=0、b=1、…、z=25），取得 Bob 的公开密钥为 $(n,e) = (74951,1033)$，加密得密文

$$\begin{aligned} c = m^e &= 11960^{1033} \\ &\equiv 11960^{2^{10}+2^3+1} \\ &\equiv \Big(\overbrace{\cdots((11960^2)^2)^2\cdots}^{10\text{次平方}}\Big)^2 \times ((11960^2)^2)^2 \times 11960 \\ &\equiv 63302 \times 47290 \times 11960 \\ &\equiv 51922 \pmod{74951}。\end{aligned}$$

- （解密）Bob 解密

$$\begin{aligned} m = c^d &= 51922^{3097} \\ &\equiv 51922^{2^{11}+2^{10}+2^4+2^3+1} \\ &\equiv \Big(\overbrace{\cdots((51922^2)^2)^2\cdots}^{11次平方}\Big)^2 \times \Big(\overbrace{\cdots((51922^2)^2)^2\cdots}^{10次平方}\Big)^2 \\ &\quad \times \Big(((51922^2)^2)^2\Big)^2 \times ((51922^2)^2)^2 \times 51922 \\ &\equiv 58071 \times 45732 \times 49264 \times 73977 \times 51922 \\ &\equiv 11960 \pmod{74951}\ , \end{aligned}$$

所以 $m = 11960 = (17,18,0)_{26} = \text{'rsa'}$

- $d \equiv e^{-1} \equiv 3097 \pmod{74951}$ 的计算过程如表 6-1 所示。

表 6-1　计算过程表

k	0	1	2	3	4	
r_k	1033	74400	1033	24	1	0
q_k		0	72	43	24	
x_k	1	0	1	72	3074	
y_k	0	1	0	1	43	

xeuclidean(1033,74951):

其中 $\begin{cases} x_{k+1} = q_k x_k + x_{k-1}\ , & x = (-1)^n x_n\ , \quad x_0 = 1, x_1 = 0 \\ y_{k+1} = q_k y_k + y_{k-1}\ , & y = (-1)^{n+1} y_n\ , \quad y_0 = 0, y_1 = 1 \end{cases}$

所以 $d = x = (-1)^4\, 3097 \equiv 3097 \pmod{74951}$

由上例可知 RSA 加密解密的计算为模指数（Modular Exponentiation）运算，计算速度远比对称密码如 DES、AES、IDEA 慢许多。

指数模运算算法的好坏，直接影响 RSA 加密解密的速度，在上例中，我们采取简易的二元算法（Binary Algorithm），顾名思义是利用其指数的二进制展开进行逐次平方计算，其 C++ 程序代码如下。

程序 6.1　计算 $\text{base}^{\text{exponent}}$（mod modulus）

```
biginteger PowerMod(biginteger base,biginteger exponent,
biginteger modulus)
//exponent>=0
{
    biginteger result=0;
    if (base==0) return result;
    else if (exponent==0) return 1;
    else if (exponent==1) return base%modulus;
    else
    {
        result=PowerMod(square(base)%modulus,
            exponent>>1,modulus);
```

```
        if (is_odd(exponent)!=1)
            return result;
        else
            return (result*base)%modulus;
    };
}
```

注 6.2 假设 base 与 modulus 皆为 l 位整数，exponent 为 b 位整数，则二元算法 PowerMod 需要最多 b−1 次模乘法与 b−1 次模平方，而每次模乘法与模平方计算复杂度为 $O(l^2)$，故二元算法 PowerMod 的计算复杂度为

$$O(l^2 \times b)\text{。}$$

6.3 RSA 的数论背景

数论被数学王子高斯（Carl Friedrich Gauss）誉为“数学女王”，20 世纪前半叶解析数论（Analytic Number Theory）大师 G.H. Hardy 在 1940 年写了一本书——《数学家的告白》（A Mathematician's Apology），很自豪地说：“真正的数学跟战争没有任何关系。还没有人发现数论有任何战争用途。”他所谓真正的数学就是指的纯数学，如数论。然而不可否认密码学对战争的重要性，自从公开密钥密码系统发展以来，数学女王数论却成为研究密码学的核心，Hardy 大师所言差矣，数论的战争用途也随着公开密钥密码系统的发展不言自明。

让我们复习另一大师 Euler 的一个定理。

定理 6.1（Euler） 令整数 a、n，其中 a 与 n 互质。则

$$a^{\phi(n)} \equiv 1 \pmod n\text{。}$$

证明：令 $\mathbb{Z}/n^{\times} := \{x \in \mathbb{Z} \mid 0 \leqslant x \leqslant n-1, \gcd(x,n)=1\}$，考虑函数

$$f_a : \mathbb{Z}/n^{\times} \to \mathbb{Z}/n^{\times}$$

$$f_a(x) = ax \pmod b \text{ 其中 } \gcd(a,n)=1\text{。}$$

- f_a 是完好定义（Well-defined）：
  可用广义辗转相除法证得，若 $\gcd(a,n)=1$、$\gcd(x,n)=1$，则 $\gcd(ax,n)=1$。
- f_a 是 1－1 函数：
  假设不是，存在相异的 x、$y \in \mathbb{Z}/n^{\times}$，使得

$$f_a(x) = ax \equiv ay \doteq f_a(y) \pmod n\text{。}$$

但 $\gcd(a,n)=1$，存在 $a^{-1} \in \mathbb{Z}/n^{\times}$，使得 $a^{-1}a \equiv 1 \pmod n$。

$$\text{所以 } x \equiv a^{-1}ax \equiv a^{-1}ay \equiv y \pmod n,$$

得 $x = y \in \mathbb{Z}/n^{\times}$，矛盾。

函数 f_a 将 $\mathbb{Z}/n^{\times}$ 元素置换，将 $\mathbb{Z}/n^{\times}$ 所有元素相乘，得

$$\prod_{x\in\mathbb{Z}/n^{\times}} x \equiv \prod_{x\in\mathbb{Z}/n^{\times}} f_a(x)$$

$$\equiv \prod_{x\in\mathbb{Z}/n^{\times}} ax \equiv a^{\phi(n)} \prod_{x\in\mathbb{Z}/n^{\times}} x \quad (\because \#\mathbb{Z}/n^{\times} = \phi(n))$$

所有 $x \in \mathbb{Z}/n^{\times}$ 都有乘法反元素，所以 $a^{\phi(n)} \equiv 1 \pmod{n}$。

注意：也可采用 Lagrange 定理证明。

Euler 定理推广了费马小定理。

定理　令整数 a 与质数 p，其中 a 与 p 互质。则

$$a^{p-1} \equiv 1 \pmod{p}。$$

定理 6.2（RSA 加密解密函数）　令 p、q 为相异质数，令 $n = pq$，令 e 为与 $(p-1)(q-1)$ 互质的整数，令 d 满足

$$ed \equiv 1 \pmod{(p-1)(q-1)},$$

令加密函数 $E(m) = m^e \pmod{n}$，解密函数 $D(c) = c^d \pmod{n}$。则对所有整数 m 皆满足

$$D(E(m)) = m \pmod{n} 且 E(D(m)) = m \pmod{n}。$$

证明：由 Euler-Phi 函数性质可知

$$\phi(n) = \phi(p)\phi(q) = (p-1)(q-1),$$

又 e 与 d 满足 $ed \equiv 1 \pmod{(p-1)(q-1)}$，故存在某整数 k 使得 $ed = k\phi(n) + 1$。考虑

$$D(E(m)) \equiv (E(m))^d \equiv (m^e)^d = m^{ed} = m^{k\phi(n)+1} \pmod{n}$$

$$E(D(m)) \equiv (D(m))^e \equiv (m^d)^e = m^{ed} = m^{k\phi(n)+1} \pmod{n}$$

考虑值 $\gcd(m,n)$：

（1）若 $\gcd(m,n) = 1$：

由 Euler 定理，得 $m^{k\phi(n)+1} \equiv (m^{\phi(n)})^k m \equiv m \pmod{n}$；

（2）若 $\gcd(m,n) = n$：

$$m^{k\phi(n)+1} \equiv 0 \equiv m \pmod{n};$$

（3）若 $\gcd(m,n) = p$ 或 $\gcd(m,n) = q$：

假设 $\gcd(m,n) = p$，此时 $\gcd(m,q) = 1$，由费马小定理得 $m^{q-1} \equiv 1 \pmod{n}$，故

$$m^{k\phi(n)+1} = m^{k(q-1)(p-1)+1} \equiv m \pmod{q},$$

另一方面，$m^{k\phi(n)+1} \equiv 0 \pmod{p}$，所以，可由中国余式定理得

$$m^{k\phi(n)+1} \equiv m \pmod{n}。$$

注 6.3　一般相信 RSA 算法是安全的，不过若 n 的因数分解已知，则该 RSA 模数 n 就会失效。在以上定理的证明中可见，虽然对所有的整数 m 都可以编码及解码，但 *m* 与 *n* 不互质且 n 不整除 m 的概率为

$$\text{Prob} = \frac{n-1-\phi(n)}{n} = 1 - \frac{1}{n} - (1-\frac{1}{p})(1-\frac{1}{q}) = \frac{1}{p} + \frac{1}{q} - \frac{2}{n},$$

此时只需要计算 $\gcd(m,n)$，就可因数分解 n。若 p、q 为 512 位质数，n 因数分解的概率约为 $\text{Prob} \approx \frac{1}{2^{511}} < 10^{-153}$，这在使用上应是安全的。

其实 RSA 的解码过程以及算法本身都可进一步改善，数论上常用的中国余式定理在此又扮演了关键的角色。

算法 6.2（RSA 修订版）　令 Alice 想将明文 m 加密成密文 c 传至 Bob，而 Bob 将密文 c 解密还原成明文 m。

（1）（密钥产生）Bob 取相异质数 p、q（保密），计算 RSA 模数 $n=pq$，将其公开，取 e 为密钥将其公开，其中 e 必须与 $\lambda(n)=\mathrm{lcm}(p-1,q-1)$ 互质，Bob 的公开密钥为 (n,e)；计算

$$d \equiv e^{-1} \pmod{\lambda(n)}、$$
$$d_p \equiv d \pmod{p-1}、$$
$$d_q \equiv d \pmod{q-1} \text{ 及}$$
$$x_q \equiv q^{-1} \pmod{p}$$

Bob 的私钥为 (n,d,d_p,d_q,x_q)。

（2）（加密）Alice 取得 Bob 的公开密钥 (n,e)，用加密函数计算

$$c=E(m)\equiv m^e \pmod{n},$$

将密文 c 传给 Bob。

（3）（解密）Bob 有两种解密计算方式：一是

$$m=D_1(c)\equiv c^d \pmod{n}$$

以及用中国余式定理加速解密

$$\begin{aligned} m &= D_{CRT}(c) \\ &\equiv x_q q(c^{d_p} \bmod p)+(1-x_q q)(c^{d_q} \bmod q) \pmod{n}。\end{aligned}$$

注 6.4 此处用中国余式定理可将 RSA 原始版本中 $ed\equiv 1 \pmod{(p-1)(q-1)}$ 修改为 $ed\equiv 1 \pmod{\lambda(n)=\mathrm{lcm}(p-1,q-1)}$，另外用中国余式定理加速解密，即解密函数 $D_{CRT}(\cdot)$，可节省大约 $\frac{3}{4}$ 的时间，大幅度节省计算成本。

密码业界先驱美国信息安全 RSA 公司提出了公开密钥密码标准（PKCS，Public Key Cryptography Standards）。注[4]此修订版 RSA 算法就是他们所提出的标准 PKCS#1。

定理 6.3 符号同算法（6.2），则对所有整数 m 皆满足

$$D_1(E(m))\equiv m \pmod{n} \text{ 且 } E(D_1(m))\equiv m \pmod{n}。$$

证明：因 $ed\equiv 1 \pmod{\lambda(n)=\mathrm{lcm}(p-1,q-1)}$，故对某整数 k，有

$$D_1(E(m))\equiv E(D_1(m))=m^{ed}=m^{k\cdot\mathrm{lcm}(p-1,q-1)+1}$$

使用费马小定理得

$$m^{ed}\equiv m \pmod{p} \text{ 且 } m^{ed}\equiv m \pmod{q},$$

由中国余式定理得 $m^{ed}\equiv m \pmod{n}$。

定理 6.4 符号同算法（6.2），则对所有整数 m 皆满足

$$D_{CRT}(E(m))\equiv m \pmod{n} \text{ 且 } E(D_{CRT}(m))\equiv m \pmod{n}。$$

证明：在此证明 $D_{CRT}(E(m))\equiv m \pmod{n}$，而 $E(D_{CRT}(m))=m \pmod{n}$ 留给读者作为习题。再次用到费马小定理，

$$D_{CRT}(E(m))\equiv x_q q((m^e)^{d_p} \bmod p)+(1-x_q q)((m^e)^{d_q} \bmod q) \pmod{n}$$

注[4] 参阅http://www.rsasecurity.com/rsalabs/pkcs。

$$\equiv m^{ed_p} \pmod q$$

$$\equiv m^{e(d+k_1(q-1))} \pmod q \qquad (\text{某整数 } k_1)$$

$$\equiv m^{1+k\cdot \mathrm{lcm}(p-1,q-1)} m^{ek_1(q-1)} \equiv m \pmod q,$$

若 $q \mid m$，此时 $D_{CRT}(E(m)) \equiv 0 \equiv m \pmod q$；同理可得

$$D_{CRT}(E(m)) \equiv x_q q((m^e)^{d_p} \bmod p) + (1 - x_q q)((m^e)^{d_q} \bmod q) \pmod n$$

$$\equiv (1-x_p p)m^{ed_p} + x_p p m^{ed_p}$$

（因为 $\gcd(p,q)=1,\ x_p p + x_q q = 1$）

$$\equiv m^{ed_p} \pmod p$$

$$\equiv m^{e(d+k_2 p)} \pmod p \quad (\text{某整数 } k_2)$$

$$m^{1+k\cdot \mathrm{lcm}(p-1,q-1)} m^{ek_2 p} \equiv m \pmod p,$$

再用中国余式定理得 $D_{CRT}(E(m)) = m \pmod p$，故得证。

注 6.5 三种不同解密函数 $D(\cdot)$、$D_1(\cdot)$、$D_{CRT}(\cdot)$ 只是计算方式不同，皆为加密函数 $E(\cdot)$ 的反函数，根据反函数的惟一性，计算结果都相同，它们其实为同一函数：

$$D(\cdot) = D_1(\cdot) = D_{CRT}(\cdot)。$$

例 6.2 Alice 想将明文'rsa cipher'加密成密文传至 Bob。

- （密钥产生）Bob 取质数 $p = 39863$、$q = 39509$，计算 RSA 模数 $n = pq = 1574947267$，加密密钥一般建议取为费马数（Fermat Number）

  $$e = F_4 = 2^{2^4} + 1 = 2^{16} + 1 = 65537,$$

  计算

  $$\lambda(n) = \mathrm{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{\gcd(p-1, q-1)}$$

  $$= \frac{39862 \times 39508}{2} = 787433948,$$

  Bob 的公开密钥为 $(n,e) = (1574947267, 65537)$，用 xeuclidean()计算解密密钥

  $$d \equiv e^{-1} = 65537^{-1} \equiv 279651573 \pmod{787433948}。$$

- （加密）Alice 的明文'rsa cipher'。约定代码为：空白=0、a=1、…、z=26，因 $[\log_{27} n] = 6$，密文每 6 个字母一组，将'rsa cipher'补为（Padding）'rsa cipherxx'。$m = (m_1, m_2) =$ 'rsa cipherxx'，

  所示 $m_1 =$ 'rsa ci' $= (18,19,1,0,3,9)_{27} = 268397478$，

  $m_2 =$ 'pherxx' $= (16,8,5,18,24,24)_{27} = 233946249$。

  取得 Bob 的公开密钥 $(n,e) = (1574947267, 65537)$，加密得密文

  $$c = (c_1, c_2) = (268397478^{65537}, 233946249^{65537}) \pmod{1574947267}$$

  $$= (1358124833,\ 1280812083)。$$

- （解密）Bob 用解密密钥 $d = 279651573$ 解密

$$m=(m_1,m_2)=(1358124833^{279651573},1280812083^{279651573})\pmod{1574947267}$$
$$=(268397478,\ 233946249)$$
$$=((18,19,1,0,3,9)_{27},(16,8,5,18,24,24)_{27})$$
$$='\text{rsa cipherxx}'。$$

例 6.3（续上例） Bob 事先计算

$$d_p\equiv d\pmod{p}=279651573\pmod{39862}\equiv 19643、$$
$$d_q\equiv d\pmod{q}=279651573\pmod{39508}\equiv 13949 \text{ 及}$$
$$x_q\equiv q^{-1}\pmod{p}=39509^{-1}\pmod{39863}\equiv 21508。$$

收到 Alice 传送的密文 $c=(c_1,c_2)=(1358124833,1280812083)$，用中国余式定理版解密函数 $D_{CRT}(c)=(x_q q(c^{d_p}\bmod p)+(1-x_q q)(c^{d_q}\bmod q)\pmod{n}$ 解密：

$$m_1=x_q q(c_1^{d_p}\bmod p)+(1-x_q q)(c_1^{d_q}\bmod q)\pmod{n}$$
$$=21508\times 39509\times(1358124833^{19643}\bmod 39863)$$
$$+(1-21508\times 39509)\times(1358124833^{13949}\bmod 39509)\pmod{1574947267}$$
$$\equiv 11508\times 39509\times 39762+(1-21508\times 39508)\times 12841\pmod{1574947267}$$
$$=268397478\pmod{1574947267}$$
$$=(18,19,1,0,3,9)_{27}$$
$$='\text{rsa ci}'。$$

同理可得

$$m_2=D_{CRT}(555061010)=233946249=(16,8,5,18,24,24)_{27}$$
$$='\text{pherxx}'。$$

所以明文为'rsa cipherxx'。

6.4 RSA 数字签名

RSA 密码系统的数字签名的实践，就是利用其加密函数与解密函数互为反函数的性质，如图 6-3 所示。Bob 将信息 m 经由解密函数加以“签名”即

$$s=D_{Bob}(m)$$

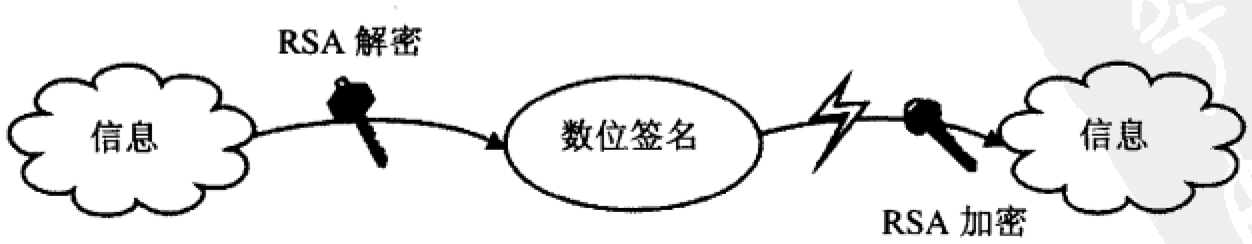


图 6-3 RSA 数字签名是可回复的数字签名法

传送给 Alice，Alice 收到 Bob 的数字签名，并取得 Bob 的加密密钥（公开密钥）d，用加密函数 $E_{Bob}(\cdot)$ 解读数字签名 s，即

$$E_{Bob}(s) = E_{Bob}(D_{Bob}(m)) = m 。$$

RSA 数字签名是一种可回复式数字签名法（Recovery Scheme），与另外的附录式数字签名法（Appendix Scheme）在其实际操作的方法上有显著的不同。

例 6.4　Bob 利用 RSA 密码系统，数字签名一简短的文件

$$m = \text{' bob cracked rsa '},$$

并传递给他所有认识的人。

- （密钥产生）Bob 的 RSA 密钥为

$$(n,e,d) = (242068116828012367991 69, 31, 19521622324306440686971);$$

其中 RSA 模数

$$n = p \times q \quad (p = 622354832383, q = 38895514943),$$

加密密钥为 e = 31，解密密钥为

$$\begin{aligned} d &= e^{-1} \pmod{\phi(n)) = (p-1)\times(q-1)} \\ &= 31^{-1} \pmod{24206811682139986451844} \\ &= 19521622324306440686971 。\end{aligned}$$

- （数字签名）假设代码空白 = 0 、a = 1 、b = 2 、…、z = 26，又

$$[\log_{27} n] = [\log_{27} 24206811682801236799169] = 15,$$

故以全文为一组编码，得代码

$$\begin{aligned} &\text{' bob cracked rsa '} \\ &= (2, 15, 2, 0, 3, 18, 1, 3, 11, 5, 4, 0, 18, 19, 1)_{27} \\ &= 2\times 27^{14} + 15\times 27^{13} + 2\times 27^{12} + 3\times 27^{10} \\ &\quad +18\times 27^{9} + 1\times 27^{8} + 3\times 27^{7} + 11\times 27^{6} \\ &\quad +5\times 27^{5} + 4\times 27^{4} + 18\times 27^{2} + 19\times 27 + 1 \\ &= 2799272500812004797 82 = m ,\end{aligned}$$

再以解密函数 $D_{Bob}(\cdot)$ 分别对此代码“数字签名”

$$\begin{aligned} s &= m^{d} \pmod n \\ &= 279927250081200479782^{19521622324306440686971} \\ &\quad \pmod{24206811682801236799169} \\ &= 17083178691394655337512 \end{aligned}$$

并将“数字签名” s = 17083178691394655337512 传给所有她想要传送的人。

- （签名还原）Alice 收到了 Bob 的“数字签名”，经过计算

$$\begin{aligned} m &= s^{e} \pmod n \\ &= 17083178691394655337512^{31} \\ &\quad \pmod{24206811682801236799169} \\ &= 279927250081200479782 \end{aligned}$$

$$=(2, 15, 2, 0, 3, 18, 1, 3, 11, 5, 4, 0, 18, 19, 1)_{27}$$
$$= \text{'bob cracked rsa'},$$

她知道此“数字签名”是真正来自于 Bob，Bob 之后就算想抵赖，也由不得他，因为“不可否认性”正是数字签名的特性。

然而 RSA 数字签名满足乘法代数性质（其他许多公开密钥密码系统都有此性质）：

$$D(m_1,m_2)=(m_1m_2)^d \mod n$$
$$\equiv (m_1^d \mod n)(m_2^d \mod n) \mod n\text{。}$$
$$\equiv D(m_1)D(m_2)\text{。}$$

这在有心人 Eve 的设计下，要伪造 Bob 的数字签名，其方法如下。

例 6.5 Eve 想设计伪造 Bob 的数字签名 $s=D(m)$，期间她花了好大功夫发现 $m\equiv m_1m_2 \mod n$，其中 m_1 与 m_2 居然都是有意义的文字的代码（虽然这在 RSA 密码系统中出现的可能性微乎其微），然而 Bob 居然毫无觉察地将信息 m_1 与 m_2 予以“数字签名”，如此

$$s=D(m)=D(m_1)D(m_2)\ ,$$

Eve 便取得 Bob 对文件 m 的数字签名，加以伪造。

若要防止如此的攻击法，一种方式就是先用 Hash 函数或重复函数（Redundancy Function）f 作用，再予以数字签名，即

$$s=D(f(m))$$

由于 Hash 函数或重复函数无法具备这样的乘法代数性质，即

$$f(m_1m_2)\neq f(m_1)f(m_2)$$

因此可有效防止此类攻击。但由于 Hash 函数的值多为 128 位或 160 位以内的固定长度，虽然防止了这类攻击，但也可能降低长密钥 RSA 密码密钥所能提供的计算安全性保证，Hash 函数的使用，还是谨慎为妙；若 Eve 懂得生日攻击（Birthday Attack），注[5]甚至可将 Hash 函数的有效计算安全长度降低一半，这样的保密程度大概只有 64 位或 80 位的有效安全长度抵抗生日攻击法。因此密码的使用，还是需要全面考虑，并非加的密码元件越多，密码就越安全。

6.5 同时进行 RSA 加密和 RSA 数字签名

在本节中，要讨论如何利用 RSA 密码技术同时进行加密以及数字签名，这样可同时实现机密性、可认证性、不可否认性。假设 Alice 欲加密明文 m 并予以数字签名，传递信息给 Bob，其中 Alice 的公钥为 (n_A,e_A) 而 Bob 的公钥为 (n_B,e_B)，标准做法如图 6-4 所示，就是 Alice 先数字签名再加密，即

$$s=m^{d_A} \pmod{n_A}$$
$$c=s^{e_B} \pmod{n_B},$$

而 Bob 则是先解密，再验证，即

$$\hat{s}=c^{d_B} \pmod{n_B}$$
$$\hat{m}=\hat{s}^{e_A} \pmod{n_A},$$

注[5] 参阅本书《生日攻击》一节。

然而只要 $n_A > n_B$，这只能保证 $m \equiv \hat{m} \pmod{n_B}$，却不能保证 $m = \hat{m}$，此时就会出现无法回复明文 m 的窘况，称为 Reblocking 问题。

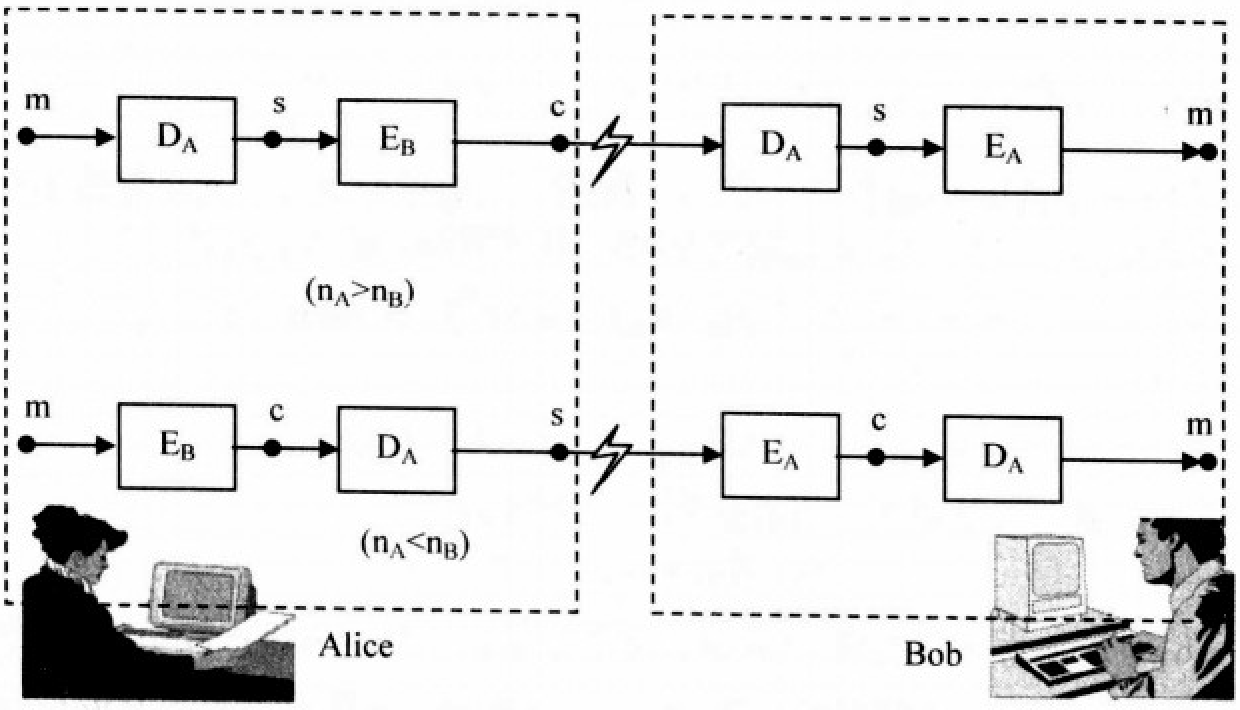


图 6-4　同时 RSA 加密和 RSA 数字签名

例 6.6（Reblocking 问题）　令 m = 4519729818307689413。而 Alice 与 Bob 的公钥分别为

$$(n_A, e_A) = (3727906129 \times 3742080863, 7) = (13950126184391309327, 7),$$
$$(n_B, e_B) = (1986525857 \times 3045502973, 5) = (6049970403434872861, 5);$$

私钥分别为

$$d_A = 7^{-1} \pmod{(3727906129-1)\times(3742080863-1)}$$
$$= 7^{-1} \pmod{13950126176921322336} = 9964375840658087383,$$
$$d_B = 5^{-1} \pmod{(1986525857-1)\times(3045502973-1)}$$
$$= 5^{-1} \pmod{6049970398402844032} = 2419988159361137613。$$

Alice 计算

$$s = m^{d_A} \pmod{n_A}$$
$$= 4519729818307689413^{9964375840658087383}$$
$$\pmod{13950126184391309327}$$
$$= 6049970403434872867,$$

$$s = s^{e_B} \pmod{n_B}$$
$$= 6049970403434872867^5 \pmod{6049970403434872861}$$
$$= 7776;$$

Bob 计算

$$\hat{s} = c^{d_B} \pmod{n_B}$$
$$= 7776^{2419988159361137613}$$
$$\pmod{6049970403434872861}$$
$$= 6,$$

$$\begin{aligned}\hat{m} &= \hat{s}^{e_A} \pmod{n_A}\\ &= 6^7 \pmod{9964375840658087383}\\ &= 279936\text{。}\end{aligned}$$

当中 $m \neq \hat{m}$，其原因就是 $s > n_B$，其发生此问题的概率为 $\frac{n_A - n_B}{n_A}$。

注 6.6 要解决 Reblocking 问题，其中一种最直接简单的方式，就是将数字签名与加密的顺序颠倒，即若 $n_A > n_B$ 时，Alice 就以先加密、再数字签名的方式处理，

$$m \to c = m^{e_B} \pmod{n_B} \to s = c^{d_A} \pmod{n_A}\text{；}$$

而 Bob 则是计算

$$s \to c = s^{e_A} \pmod{n_A} \to m = c^{d_B} \pmod{n_B}\text{。}$$

其他的解决方案，则是每个使用者必须有两组钥匙

$$(n_{A_1}, e_{A_1}, d_{A_1})\text{、}(n_{A_2}, e_{A_2}, d_{A_2})\text{，}$$

其中 $n_1 < n_2 < n_3$，而 n_c 为一组临界值，发送方加密，取本人模数较小的私钥 (n_{A_1}, e_{A_1}) 先数字签名，而后再用对方较大模数的公钥 (n_{B_2}, e_{B_2}) 加密，这样也可避开 Reblocking 问题，但这样一来，便倍增了所需的密钥数目。

例 6.7（续上例） 由于 $n_A > n_B$，Alice 就用先加密、再数字签名的方式处理：

$$\begin{aligned}c &= m^{e_B} \pmod{n_B}\\ &= 4519729818307689413^5 \pmod{6049970403434872861}\\ &= 3160066570610330395\text{，}\end{aligned}$$

$$\begin{aligned}s &= c^{d_A} \pmod{n_A}\\ &= 3160066570610330395^{9964375840658087383}\\ &\quad \pmod{13950126184391309327}\\ &= 2714048735005420620\text{，}\end{aligned}$$

Bob 先验证，再解密：

$$\begin{aligned}c &= s^{e_A} \pmod{n_A}\\ &= 2714048735005420620^7 \pmod{13950126184391309327}\\ &= 3160066570610330395\text{，}\end{aligned}$$

$$\begin{aligned}m &= c^{d_B} \pmod{n_B}\\ &= 3160066570610330395^{2419988159361137613}\\ &\quad \pmod{6049970403434872861}\\ &= 4519729818307689413\text{。}\end{aligned}$$

Reblocking 问题如此获得解决。

6.6 RSA-129 挑战与因数分解

RSA 密码系统在本质上是双质数密码（Bi-prime Cipher），在实际操作中，必须关注以下

两个质数问题：

（1）RSA 的模数 $n = pq$，其中 p、q 为相异质数，如何找到质数 p、q？又如何判断所找到的数是质数？

（2）在 RSA 密码系统中 RSA 模数 n 是公开的，如果有人能因数分解 $n = pq$，任何以此模式 n 设计的 RSA 密码，都可破解，因此模数 n 不能轻易被因数分解成了 RSA 密码系统。在实际使用时最需要考虑的问题；数论学家在因数分解问题上的进展程度，成了 RSA 密码是否能继续使用的关键。

在此，我们将试图考虑第二个质数问题，即因数分解问题。要解释 RSA 密码系统与因数分解的关联性，可由以下范例中一窥端倪。

例 6.8 Bob 的 RSA 公开密钥为

$$(n,e) = (782741, 7)。$$

攻击者 Eve 截收到 Alice 传给 Bob 的密文 144310，并知道 Alice 与 Bob 的通信代码为：

$$a = 0、b = 1、c = 2、\cdots、z = 25。$$

计算 $[\log_{26} n] = [\log_{26} 782741] = 4$，知道每 4 个字母为一组代码。如果 Eve 知道 n 的质因数分解，即知道是何质数 p、q 使得 $pq = 782741$，就可计算 $\phi(n) = (p-1)(q-1)$，再以广义辗转相除法或 xeuclidean()程序计算得到解密密钥

$$d \equiv 7^{-1} \mod \phi(n),$$

便可还原明文的代码

$$m \equiv 144310^{d} \mod \phi(782741)。$$

如何找到质因数 p、q 使得 $pq = 782741$？Eve 试着用从 2 到小于等于 $[\sqrt{782741}] = 884$ 的质数试除（共计有 153 个质数），在第 150 个质数时发现

$$782741/863 = 907 = q,$$

因此 $\phi(n) = (p-1)(q-1) = 862 \times 906 = 780972$。以广义辗转相除法得

$$d = 7^{-1} \equiv 223135 \ (\text{mod } \phi(n) = 780972),$$

所以明文代码为

$$144310^{223135} \equiv 126037 \ (\text{mod } 782741) = 126037,$$

而

$$\begin{aligned} 126037 &= 4847 \times 26 + 15 \\ &= (186 \times 26 + 11) \times 26 + 15 \\ &= ((7 \times 26 + 4) \times 26 + 11) \times 26 + 15 \\ &= (7, 4, 11, 15)_{26}。\end{aligned}$$

因此明文为“help”。

从上例中，我们发现，在已经有质数表的情况下，Eve 还需要计算 120 次除法，方可分解 6 位数 $n = 782741$，即使是如此小的 6 位数，质因数分解尚且需要如此因此多的计算，质因数分解是公认非常困难的。而 RSA 密码的安全性基于质因数分解，因此能够将 RSA 模数 n 质因数分解，就能破解任何以此模式 n 为公开密钥的 RSA 密码；然而能破解 RSA 密码，却不一定就能质因数分解 RSA 模数 n（质因数分解 RSA 模数 n 是破解 RSA 密码的充分条件，而不是必要条件）。

用因数分解法在计算 l 位整数 n，以目前所用的算法来看是困难的，所有的算法，都无法在多项式时间内完成，以二次筛法（Quadratic Sieve Method）为例，其计算复杂度为

$$e^{(1+o(1))\sqrt{\ln n \ln\ln n}} = e^{O(\sqrt{l \ln l})},$$

而一般的代数域筛法（Algebraic Number Field Sieve Method）也须

$$e^{(1.923+o(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}} = e^{O(l^{1/3}(\ln l)^{2/3})}。$$

试图破解 RSA 密码，就意味着尝试质因数分解 RSA 密码公开密钥的 RSA 模数 n，如图 6-5 所示，表 6-2 所示为历史上人类质因数分解的部分记录。

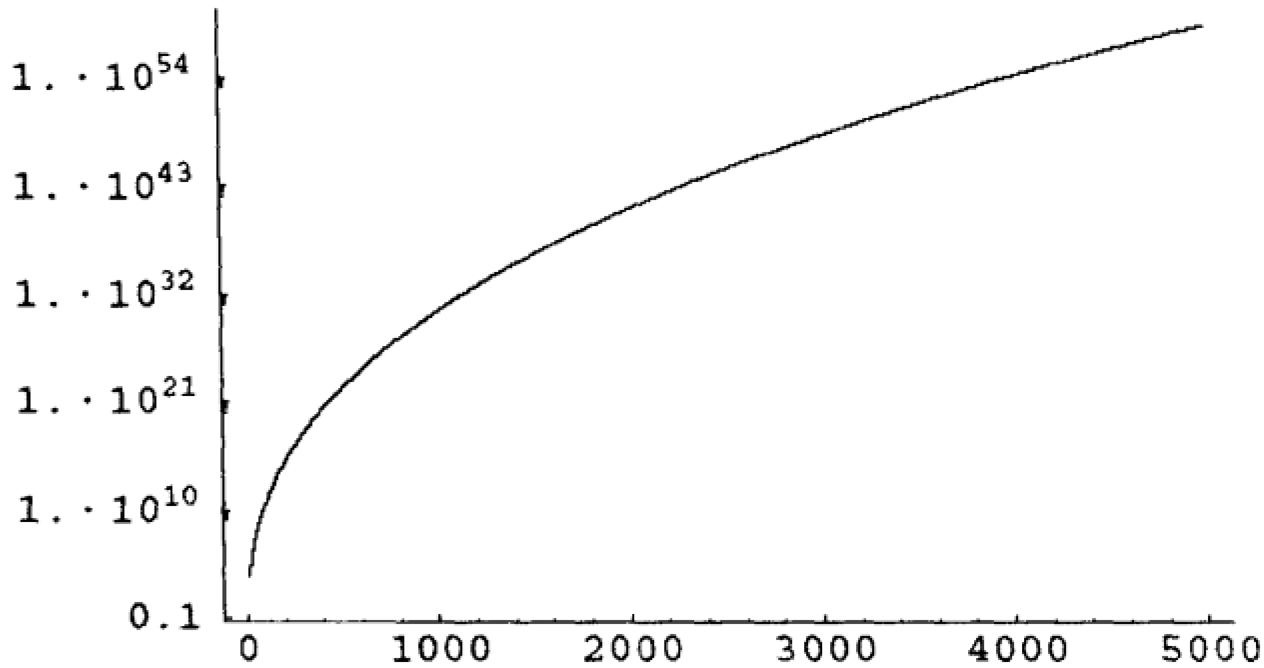


图 6-5　用代数域筛法因数分解不同位密钥的 RSA 模数所需的计算次数

表 6-2　质因数分解的记录

年份	十进制的位数	备注
1964	20	
1974	45	
1983	69	$2^{251}-1$ 用二次筛法
1989	106	用二次筛法
1994	129	RSA-129 用二次筛法
1996	130	RSA-130 用代数数体筛法
1999	155	RSA-155 用代数数体筛法

值得一提的是 RSA-129 挑战破解的经过，在 Rivest、Shamr、Adleman 的 RSA 算法初次发表之际，就在《科学美国人》中提出模数为 129 的十进位数

$$\begin{aligned} n = {} & 114381625757888867669235779976146612010 2 \\ & 1829672124236256256184293570693524573389 \\ & 7830597123563958705058989075147599290026 \\ & 879543541, \end{aligned}$$

加密密钥 e = 9007 的密文

$$c = 96869613754622061477140922254355882905759991124574319874695120930816298225145708356931476622883989628013391990551829945157815154\ ,$$

悬赏 100 美金给任何能在公元 1982 年 4 月 1 日愚人节之前破解的人。以当时的质因数分解技术，他们推算应费 4×10^{16} 年方可破解成功。1982 年愚人节，这项 RSA-129 挑战未能破解。但在 1994 年，多位数论学家 Atkins、Graff、Lenstra 以及 Leyland 改良了因式分解的技巧，提出改良版的二次筛法算法，结合全球 600 多人的团队用 1600 部电脑执行了 8 个月，当中尝试以高斯消去法（Gauss Elimination）找出了数十万项变量的关联，终于成功地将质因数 n 分解成 $n = p\times q$ ，即

$$p = 3490529510847650949147849619903898133417764638493387843990820577\ ,$$

$$q = 32769132993266709549961988190834461413177642967992942539798288533\ ,$$

如此以广义辗转相除法计算解密密钥

$$\begin{aligned} d &= e^{-1} \pmod{\phi(n) = (p-1)(q-1)} \\ &= 106698614368578024442868771328920154780709906633937862801226224496631063125911774470873340168597462306553968544513277109053606095 \end{aligned}$$

明文数字代码为

$$\begin{aligned} m &= c^{d} \pmod n \\ &= 200805001301070903002315180419000118050019172105011309190800151919090618010705\ , \end{aligned}$$

以代码 $a = 01$ 、 $b = 02$ 、…、空白键 = 00 解读为

```
'the magic words are squeamish ossifrage'
```

即密语是一种神经质鹰（squeamish ossifrage 为一种生长在北美的易受惊吓的鹰），总共计算次数为 5000 MIPS-年。另外公布在 RSA 信息安全公司首页的 RSA-155 挑战也在 1999 年破解，计算次数为 8400 MIPS-年，用的是代数数域筛法；而在 Singh 所著的码书（The Code Book）附录中所录的第十个悬赏密文，也是 155 十进位的 RSA 密文，也被瑞典一个研究团队于 2000 年首次破解。

6.7　二次筛法 Pollard 的 p－1 法

破解 RSA-129 采用的是某种版本的二次筛法，用以因数分解 RSA 模数 n，二次筛法在解析数论中是一种常用的标准方法，它的基本想法，如同许多其他的因数分解法一样，是要找出正整数 x、y，使得

$$x^2 \equiv y^2 \mod n$$

并且满足

$$x \not\equiv \pm y \mod n,$$

如此 n 是数 $x^2 - y^2 = (x+y)(x-y)$ 的因数，却非 $x+y$ 或 $x-y$ 的因数，故其 $\gcd(x-y,n)$ 或 $\gcd(x+y,n)$ 均为 n 的因数；如此便可将 n 因数分解。

例 6.9（Fermat 因数分解） 与二次筛法一样，Fermat 因数分解就是利用式子

$$n = x^2 - y^2 = (x+y)(x-y)。$$

令 $n = 295927$。直接计算 $n+1^2$、$n+2^2$、…、$n+k^2$ 直至 $n+k^2$ 为完全平方数为止。在本例中，

$$295927 + 3^2 = 295936 = 544^2,$$

因此可得因数分解

$$295927 = 544^2 - 3^2 = (544+3)(544-3) = 547 \times 541。$$

因有 Fermat 因数分解之故，在 RSA 密码系统中的质数 p 与 q 的选取，一般建议 $|p-q|$ 不宜太小，故 p 与 q 长度宜有数字之差。

6.7.1 二次筛法

反观二次筛法是针对较一般的整数，但若应用到仅十余位以下的整数 n，就显得非常不合适（因为此时，就算是用试除法（Trial Divison），即用所有小于等于 $[\sqrt{n}]$ 的质数，逐次试除，也不算慢）。以下的例子，其最佳的因数分解方法，当然不是用二次筛法，在此只是用来说明二次筛法的应用。

例 6.10 令 $n = 9671$ 为正整数，又知 $x = 672606$、$y = 31850$，且 $n = x^2 - y^2$ 得 $x - y = 640756$，$x + y = 704456$，得 $\gcd(x-y,n) = \gcd(x+y,n) = 509$，故 509 为 9671 的因数。

如何找出 x 与 y？其中一个方法就是如同 Fermat 因数分解，以 RSA 模数 $n = pq$ 为例，若 $|p-q|$ 为长达数十位的整数，也难以执行；另一种方法，就是考虑联立平方同余等式，再求解。

例 6.11（续上例） 令 $n = 9671$，$m = [\sqrt{9671}] = 98$，令函数

$$f(x) = (x+m)^2 - n = (x+98)^2 - 9671。$$

“很幸运地”，有以下联立方程组：

$$f(1) = 130 = 2 \times 5 \times 13$$

$$f(-12) = -2275 = -1 \times 5^2 \times 7 \times 13$$

$$f(-19) = -3430 = -1 \times 2 \times 5 \times 7^3$$

可得联立同余方程组：

$$99^2 \equiv 2 \times 5 \times 13 \mod 9671$$

$$86^2 \equiv -1 \times 5^2 \times 7 \times 13 \mod 9671$$

$$79^2 \equiv -1 \times 2 \times 5 \times 7^3$$

将以上联立同余方程组各式相乘可得

$$(99 \times 86 \times 78)^2 = (672606)^2 \equiv (2 \times 5^2 \times 7^2 \times 13)^2 = 31850 \mod 9671,$$

此即 $x = 672606$、$y = 31850$。

由上例可见，我们其实是利用同余式

$$f(x) \equiv (x+m)^2 \mod n$$

“导出”了想要的平方同余式；另外一方面，还找出特殊的 x 使得 f(x) 仅有“小”质因数，然而实际操作上，则是利用二元筛法“筛选”出适当的同余线性联立方程组，再以线性代数的标准方法——高斯消去法求得答案。在继续进一步讨论二元筛法前，先介绍在计算数论上常用的概念。

定义 6.2（B-smooth）　令 B 为某若干质数与 −1 所形成的集合。令 x 为一整数：称 x 为 B-smooth $\Leftrightarrow$ x 的质因子均在集合 B 中。

为方便起见，以质数基底

$$B_n := \{p \mid p \leqslant n, p\ 为质数或\ p = -1\}$$

表示所有小于等于 n 的质数 p 或 p = −1 的集合，二元筛法就是利用适当的筛区间（Sieving Interval）

$$S = \{-C, -C+1, -C+2, \cdots, 0, 1, 2, \cdots, C-1, C\}$$

找出所有的整数 $s \in S$ 使得 f(s) 为 B_n-smooth。

注 6.7　分别以长度为 50、100 的十进位数而言，其所应选取的质数基底应为 B_{3000}、B_{51000}，而筛区间的 $C = 2\times10^5$、1.4×10^7。

例 6.12（续上例）　令 n = 9671，选质数基底

$$B = \{-1, 2, 5, 7, 13\}$$

即可（因 $x^3 \equiv 9671 \pmod{11}$ 在 $\mathbb{Z}/11$ 中无解，其 Legendre 符号 $\left(\frac{9671}{11}\right) = -1$，可不包含在内），而筛区间取

$$S = \{-20, -19, \cdots, -1, 0, 1, 2, \cdots, 19, 20\}。$$

此时，需要逐一检查 f(x)（$x \in S$）是否为 B-smooth。此时用 $p \in B$ 逐一试除可知，非 B 的项皆可“筛除”，可得结果如表 6-3 所示。

表 6-3　结果表

x	f(x)	质因数分解
1	130	$2\times5\times13$
2	−455	$-1\times5\times7\times13$
−9	−1750	$-1\times2\times5^3\times7$
−12	−2275	$-1\times5^2\times7\times13$
−19	−3430	$-1\times2\times5\times7^3$

由此可得表 6-4 结果。

表 6-4　表 6-3 结果

x	1	−2	−9	−12	−19
f(x)	130	−455	−1750	−2275	−3430
−1 的幂次	0	1	1	1	1
2 的幂次	1	0	1	0	1

续表

5 之幂次	1	1	3	2	1
7 之幂次	0	1	1	1	3
13 之幂次	1	1	0	1	0

可得矩阵等式

$$\begin{bmatrix}0&1&1&1&1\\1&0&1&0&1\\1&1&3&2&1\\0&1&1&1&3\\1&1&0&1&0\end{bmatrix}\begin{bmatrix}\lambda_1\\\lambda_2\\\lambda_3\\\lambda_4\\\lambda_5\end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix},$$

因平方同余式的取得在于其各项因数的幂次必为偶数，所以只需考虑模 2 矩阵运算（即在 $\mathbb{F}_2$ 上的 XOR 运算），运用高斯消去法解：

$$\begin{bmatrix}0&1&1&1&1&0\\1&0&1&0&1&0\\1&1&1&0&1&0\\0&1&1&1&1&0\\1&1&0&1&0&0\end{bmatrix}\xrightarrow{P_{15}}\begin{bmatrix}1&1&0&1&0&0\\1&0&1&0&1&0\\1&1&1&0&1&0\\0&1&1&1&1&0\\0&1&1&1&1&0\end{bmatrix}$$

$$\xrightarrow{E_{31}\circ E_{21}\circ E_{45}}\begin{bmatrix}1&1&0&1&0&0\\0&1&1&1&1&0\\0&0&1&1&1&0\\0&1&1&1&1&0\\0&0&0&0&0&0\end{bmatrix}\xrightarrow{E_{42}\circ E_{23}}\begin{bmatrix}1&1&0&1&0&0\\0&1&0&0&0&0\\0&0&1&1&1&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix},$$

（其中 P_{ij} 表示交换第 i、j 列（Row），E_{ij} 表示将第 j 列加至第 i 列）不需要若干次列运算（Row Operation）可得到解

$$\begin{cases}\lambda_1=s\\\lambda_2=0\\\lambda_3=s+t\quad(\text{其中 } s、t\in\mathbb{F}_2),\\\lambda_4=s\\\lambda_5=t\end{cases}$$

代入其中一组特殊解 $(\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5)=(1,0,0,1,1)$，得

$$\begin{aligned}&(98+1)^2(98-12)^2(98-19)^2\\&\equiv130\times(-2275)\times(-3430)\mod 9671\\&=(2\times5\times13)\times(-1\times5^2\times7\times13)\times(-1\times2\times5\times7^3)\\&=(2\times5^2\times7^2\times13)^2,\end{aligned}$$

于是，x 与 y 皆可得解。

数论上用来因数分解的方法很多，如先前所提到的 Fermat 因数分解、二次筛法，另外同样是筛法的还有代数数域筛法以及函数域筛法（Function Field Sieve Method），最大的差别就在于 f(x) 的选取不同。

6.7.2 Pollard 的 p－1 法

另外在 RSA 提出之际，就曾建议质数 p 与 q 的选取，应该为“强质数”，注[6]然而在稍后的年代 RSA 的原创者之一 R 与他人合作了一篇文章，注[7]文中却又反驳 p、q 为“强质数”是不必要的。两篇文章最主要的依据都是当代的因数分解技术，要用“强质数”是因为 Pollard 的 p－1 法注[8]及其类似的 Williams 的 p＋1 法；注[9]而不用“强质数”，却是因为椭圆曲线因数分解（Elliptic Curve Factorization），注[10]因椭圆曲线因数分解的特性是可平行处理计算，然而就其算法本质而言，都可算是 Pollard 的 p－1 法在椭圆曲线的推广。而以 Pollard 的 p－1 法，若考虑在 RSA 模数 $n = pq$ 上因数分解，倘若某较大的自然数 k 可整除 p－1，此时必存在某自然数 m，使得 $k = m(p-1)$，依据费马小定理可得

$$a^k \equiv (a^{p-1})^m \equiv 1^m \equiv 1 \mod p$$

当 $\gcd(a,p)=1$。因此可得

$$p \mid a^k - 1。$$

若 $n \mid a^k - 1$，则 $\gcd(a^k - 1, n)$ 为 n 的因数。其算法可整理如下：

算法 6.3（Pollard 的 p－1 法）

输入≫ 非质数的正整数 *n*

输出≪ n 的某因数

```
Pollard_P-1(n)
{
initial:
    input>>B;
    a=2;//选取正整数 a>1，通常是取 a=2
//计算 a=(a^k-1) mod n
//其中 k=B!
    for (i=2;i<=B;i++)
        a=PowerMod(a,i,n);//PowerMod(a,i,n)=a^i mod n
    a--;
    factor=gcd(a,n);//计算 factor=gcd(a^k-1,n)
    if (factor>1 && factor<n)
        return factor;
```

注[6] 参见本书“公开钥密码与质数”一节。

注[7] R.Rivest 与 Robert D.Silverman 的“Are ‘Srrong’Primes Needed for RSA?” IACR Cryptology,(2001)。

注[8] J.M.Pollard 的“Theorems on factorization and primality testing.” Proccedings Cambridge Philosphical Society,76,P.521-528(1974)。

注[9] H.C.Williams 的“Ap+1 method for factoring” Mathematics of Computation,39(159)P.225-234(1982)。

注[10] 参见本书“椭圆曲线因数分解”一节。

```
    else
            {
        output<<"B 值太小";
        goto initial;
    };
}
```

注意：对所给定 B，可更经济地选取数值 k 计算

$$k = k(B) := \prod_{g:\text{prime},\, g^e \leqslant B} g^e 。$$

这种 k 的选取方法可行，是因为所有最高质因数幂次方 $q^e \leqslant B$ 且整除 $p-1$ 时，$p-1$ 为 k 的因数。

例 6.13 令 $n = 1241143$，取 $B = 13$ 则

$$k = \prod_{g:\text{prime},\, q^e \leqslant B} q^e = 2^3 \times 3^2 \times 5 \times 7 \times 11 \times 13 = 360360 ,$$

计算

$$\begin{aligned} \gcd(2^{360360} - 1, n) &= \gcd((2^{360360} - 1) \bmod n, n) \\ &= \gcd(32273, 1241143) \\ &= 547 = p \quad (\text{因为 xeuclidean()}) \end{aligned}$$

因此 $n/p = 1241143/547 = 2269$， n 的质因数分解为 $1241143 = 547 \times 2269$。

6.8 利用 RSA 私钥因数分解

任何 RSA 密码的使用者若知道自己的 RSA 公钥 (n,e) 以及 RSA 私钥 (n,d)，其实不难因数分解模数 $n = pq$，在此我们假设所使用的 RSA 密码是原始版本，即 e 与 d 满足

$$ed \equiv 1 \pmod{\phi(n) = (p-1)(q-1)} 。$$

其主要的想法如下：由于 $ed \equiv 1 \pmod{(p-1)(q-1)}$，故存在某整数 $m \in \mathbb{Z}$ 使得

$$ed - 1 = m(p-1)(q-1) 。$$

随机选取一个正整数 x （$2 \leqslant x \leqslant n$），若 $\gcd(x,n) > 1$，那就太幸运了，由于 Euler 定理，必然

$$x^{ed-1} \equiv 1 \pmod n$$

计算 $x^{ed-1} \pmod n$ 的一个平方根，

$$y_1 = \sqrt{x^{ed-1}} = x^{\frac{ed-1}{2}} \pmod n ,$$

（其中 $ed-1$ 必为偶数），可得同余式

$$y_1^2 - 1 \equiv 0 \pmod n ,$$

若很幸运地，$y_1 \not\equiv \pm 1 \pmod n$，那么可计算

$$\gcd(y_1 - 1, n)$$

得到 n 的一个因数。

若 $y_1 \equiv -1 \pmod n$，那么就要重换 x；若 $y_1 \equiv 1 \pmod n$，那可能还有机会，要看 $\frac{ed-1}{2}$ 是否为偶数，若否，也要重换 x；若是，计算

$$y_2 = \sqrt{y_1} \equiv x^{\frac{ed-1}{4}} \pmod n,$$

可得同余式

$$y_2^2 - 1 = y_1 - 1 = 0 \pmod n;$$

若 $y_2 \not\equiv \pm 1 \pmod n$，计算

$$\gcd(y_2 - 1, n),$$

可得 n 的一个因子，若否，看 $y_2 \equiv -1 \pmod n$ 或是 $y_2 \equiv 1 \pmod n$，依情况重换 x 或继续计算 y_3。如此程序一直进行至 $\frac{ed-1}{2^t}$ 不再被 2 整除，若还不行，就要换 x。如此计算的过程，在任意随机选取一次 x 成功的概率至少是 1/2，因此在 k 次选取不同 x 能成功的概率至少是 $1-\left(\frac{1}{2}\right)^k$，这类算法，是概率式的可能会进行无穷次循环计算，但找到的答案一定正确，有这种性质的算法也称作 Las Vegas 算法。我们将此算法整理如下。

算法 6.4

输入≫RSA 参数 e、d、n，其中 $ed \equiv 1 \pmod{\phi(n)}$

输出≪ n 之一质因数

```
factor(n,e,d)
{
initial:   x=Ramdem_Integer(2,n-1);
    if(gcd=gcd(x,n)>1)
        return gcd;
    else
        for(s=(ed-1)/2; is_odd(s); s/=2)
        {
            y=(x^s)%n;
            if(y==1)
                continue;
            else
                if(y==n-1)
                    break;
            else
                return gcd(y-1,n);
        };
    goto initial;
};
```

例 6.14　假设 Alice 的 RSA 密钥为

$$\begin{aligned}(n,e,d) = (&29199619490932896138040207735768920167,\\ &27335382451787132794462673391180552715 7,\\ &60102512763177539335068764336403641333)。\end{aligned}$$

Alice 自行因数分解 n 如下：

（1）随机找一个整数

$$x = 60541520037091837230139960944836511430,$$

很正常地 $\gcd(x,n)=1$。

（2）计算

$$s_1 = \frac{ed-1}{2}$$
$$= 8214625863474377432813546698865260701799025836995560747647339280085209590140,$$

计算

$$y_1 = x^{s_1} \pmod n = 1,$$

（3）计算

$$s_2 = \frac{ed-1}{4}$$
$$= 4107312931737188716406773349432630350899512918497780373823669640042604795070,$$

计算

$$y_2 = x^{s_2} \pmod n$$
$$= 162916502317468198293896370228560679357,$$

（4）因此其中一个因数为

$$p = \gcd(y_2 - 1, n) = 17623662805242429011,$$

而另一个因数为

$$q = n/p = 16568611344746855197。$$

6.9 RSA 密码系统使用的注意事项

RSA 密码系统是目前所广泛使用的密码计算中安全性最高的密码系统之一，但这并不表示 RSA 密码可随意使用。在实际应用上，也应注意若干事项，其中参数以及钥匙(p,q,e,d)的选取也有若干须要注意的事项。在本节中，我们将对使用中所可能引发的问题在以下的范例以及注解中讨论。

例 6.15（通用模数协议错误，Common Modulus Protocol Failure） 令 m 为明文，假设 Alice 想将此明文分别利用 RSA 加密传递给 Bob 以及 Carl，她取得的公开密钥分别是 (n,e_B) 以及 (n,e_C)，当中两组 RSA 模数 n 是相同的。而攻击者 Eve 分别截收到 Alice 寄给 Bob 的密文 $c_B = m^{e_B} \bmod n$ 以及给 Carl 的密文 $c_C = m^{e_C} \bmod n$，此时 Eve 只需要计算，便可以破解密文，为什么？首先两把加密密钥 e_B 与 e_C 互质的概率极高，即 $\gcd(e_B,e_C)=1$，不互质的概率微乎其微。此时 Eve 只需要使用广义辗转相除法或程序 xeuclidean()，便可求出整数 s 、 t 使得

$$e_B s + e_C t = 1$$

如此

$$c_B^s c_C^t = (m_B^{e_B})^s (m_C^{e_C})^t = m^{e_B s + e_C t} \equiv m \pmod n 。$$

例 6.16（低幂次 e 攻击）　有人为加快 RSA 加密速度，喜欢使用很小的加密密钥 e，甚至使用 $e=3$，以此为例，假设 Eve 已截收到相同明文的不同三份密文如下：

$$\begin{cases}(n_1,e)=(443929,3), & C_1=217560\\(n_2,e)=(346781,3), & C_2=51086\\(n_3,e)=(779917,3), & C_3=363662\end{cases}$$

由中国余式定理，只需要计算 x_1、x_2、x_3 满足

$$\begin{cases}x_1n_2n_3\equiv 1 \mod n_1\\x_2n_1n_3\equiv 1 \mod n_2\\x_3n_1n_2\equiv 1 \mod n_3\end{cases}$$

解得

$$(x_1,x_2,x_3)=(217560, 51086, 363662),$$

因此可令

$$c = c_1x_1n_2n_3 + c_2x_2n_1n_3 + c_3x_3n_1n_2 \pmod{n_1n_2n_3} \equiv 2002138753532653,$$

三次开根号并非难事，又知通信代码为：$a=0$、$b=1$、$c=2$、…、$z=25$，故

$$\begin{aligned}m &= \sqrt[3]{2002138753532653} = 126037\\&= (7,4,11,15)_{26}\\&= \text{'help'} 。\end{aligned}$$

由上例可知，要避免使用很小的加密密钥 e，一般而言，相同加密密钥 e 对 k 个不同的明文 m_i 有线性关系 a_im+b_i（a_i，b_i 已知），在不同的公钥上加密不同公钥，若 $k \geqslant \frac{e(e+1)}{2}$，就有方法攻击，而加密在不同公钥相同明文 m 之上就可用如同上例方法攻击。

例 6.17（循环攻击，Cycling Attack）　循环攻击就是将密文 c 多次用加密函数作用，直至

$$e^{e^k} = \overbrace{((\cdots(e^e)^e\cdots)^e)^e}^{k次} = c,$$

此时

$$m = e^{e^{k-1}},$$

当 $\gcd(p-1,q-1)$ 大时循环攻击特别奏效。而广义循环攻击，则是将密文 c 多次用加密函数作用，直至

$$\gcd(n, c^{e^u}-c) > 1 。$$

当 $p-1$ 与 $q-1$ 都只有小质因数时，广义循环攻击特别奏效；p、q 为“强质数”也可防止此类攻击。

综合以上各项以及以上各节所述，在使用 RSA 密码系统时，应注意以下事项：

（1）绝对不可泄漏 RSA 密码系统的任何一个参数：p、q、$\phi(n)$，这与公开自己的私钥无异。

（2）使用 RSA 数字签名，建议先使用 Hash 函数，以防止利用其乘法代数性质伪造数字

签名。

（3）绝对不要使用相同的 RSA 模数犯下相同模数协议的错误。

（4）加密密钥 e 的长度不得少于 RSA 模数 n 长度的 $\frac{1}{4}$。注[11]

（5）另外因为数论家在因数分解上的进展，为防止 n 被质因数分解，也应考虑以下建议：

1）p、q 的大小，长度应有数字之差。

2）p、q 建议采用“强质数”（不少专家仍有不同意见）。

3）n 的长度至少应为 1024 位以上（上个世纪曾建议的 512 位的长度，如 RSA-155 也于世纪交替前被破解）。

4）$\gcd(p-1, q-1)$ 的值越小越好。

是否只要遵循了上例应注意事项，使用 RSA 就万无一失？由于 RSA 已发展超过 20 年，还有许多相关密码分析技术，我们不能都介绍，注[12]然而两个发展方向值得我们注意：

- 一是计算数论的发展，可能会也将会提出更有效的因数分解的算法，目前就笔者所知，印度人 Agrawal、Kayal 以及 Saxena 师徒三人在 2002 年证明了，他们能在多项式时间内判断任何整数是否为质数，这虽然与质因数分解有一些距离，但值得期待。虽同是数论上的重大结果，他们所采用的证明方法，出人意料的竟是一般数学家擅长的基础数论，这与 Wiles 证明 Fermat's Last Theorem 大量使用代数几何（Algebraic Geometry）、交换代数（Commutative Algebra）的方法有极大的差别。
- 另一方面值得注意的是量子电脑（Quantum Computer）的发展可能足以颠覆现有的信息科技。目前已有诸多实验室正在研发，目前所知，已能完成 7 位，待技巧成熟以后，借助量子电脑平台的量子计算（Quantum Computation），将可在多项式时间内因数分解，如此的可能性应该不是缘木求鱼。

6.10 Wiener 低幂次 d 攻击

由于 Wiener 低幂次 d 攻击，不得不非常小心，在 RSA 密钥的选取上，私钥值 d 的长度必须大于 RSA 模数 n 值长度的 1/4，这对试图以大幅度降低 d 值来增加计算速度的想法，无异于一个坏消息，因为只要私钥值 d 太短或不够长，就会暴露 RSA 模数 n 随时被因数分解的危机。Wiener 的攻击法，主要是用到连分数，而有关连分数的基本概念，可参阅本书“基础数论”一章，注[13]当中实数 x 与其连分数的各项收敛值 $\frac{c_k}{d_k}$ 的关系式为：

定理 6.5 令 $\frac{c}{d}$（且 $\gcd(c,d)=1$）为实数 x 的某项连分数的收敛值 ⇔

注[11] 参见下一节“Wiener 低幂次 d 攻击”。

注[12] 有兴趣的读者建议阅读 Dan Boneh 的“Twenty years of Attacks on the RSA Cryptosystem” Notices of AMS,46,P.203-213(1999)。

注[13] 参见本书“连分数”一节。

$$\left|x-\frac{c}{d}\right| \leqslant \frac{1}{2d^2}。$$

Wiener 攻击法主要是基于以下事实注[14]

定理 6.6（Wiener 攻击）　令 RSA 密码系统中各参数满足下列条件：

$$n = pq，其中 q < p < 2q$$

$$ed \equiv 1 \pmod{\phi(n) = (p-1)(q-1)}$$

$$d < \frac{1}{3}\sqrt[4]{n}。$$

则存在多项式时间算法可因数分解 RSA 模数 n。

证明：假设 $e < \phi(n)$（这是合理假设，大部分 RSA 密码系统皆如此）存在某整数 k 使得

$$ed - k\phi(n) = 1，$$

得

$$\left|\frac{e}{\phi(n)} - \frac{k}{d}\right| = \frac{1}{d\phi(n)}。$$

因 $\phi(n) \approx n$，故

$$|n - \phi(n)| = |p + q - 1| < 3\sqrt{n}，$$

注意，值 $\frac{k}{d}$ 未知，但是值 $\frac{e}{n}$ 已知，且应该非常逼近 $\frac{k}{d}$。此时

$$\begin{aligned}\left|\frac{e}{n} - \frac{k}{d}\right| &= \left|\frac{ed - nk}{dn}\right| \\ &= \left|\frac{ed - k\phi(n) - nk + k\phi(n)}{dn}\right| \\ &= \left|\frac{1 - k(n - \phi(n))}{dn}\right| \\ &\leqslant \left|\frac{3k\sqrt{n}}{dn}\right| \\ &= \frac{3k}{d\sqrt{n}}。\end{aligned}$$

由假设 $d < \frac{1}{3}\sqrt[4]{n}$ 可得

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{1}{2d^2}，$$

因 $\gcd(k,d) = 1$，故 $\frac{k}{d}$ 必为 $\frac{e}{n}$ 的连分数的某项收敛值，而 $\frac{e}{n}$ 化为连分数，只需 O(n) 次乘除法计算；要求得正确的 d 值，只要检查连分数的收敛值 $\frac{c_j}{d_j}$ 是否满足

注[14] M.Wiener 的 " Cryptanalysis of short RSA secret exponents " IEEE Trans.Inform.Theory，36P.553-558(1990)。

$$d_j e - 1 \equiv 0 \pmod{c_j},$$

因为若 $d = d_j$ 则 $c_j = k$，此时

$$ed - 1 \equiv 0 \pmod{k}$$

$$\Leftarrow \quad ed = 1 + k\phi(n)\ 。$$

而值 $\dfrac{d_j e - 1}{c_j}$ 即 $\phi(n)$，此时 n 的质因数即一元二次方程式

$$x^2 - (n - \phi(n) + 1)x + n = 0$$

的解。

以下为 Wiener 攻击法的算法。

算法 6.5（Wiener 攻击算法）

输入≫RSA 密码的参数 e、d、n

输出≪解密密钥值 d 以及 n 的质因子 p、q

```
Wiener(n,e,d)
{
//利用修订版的辗转相除法求 e/n 的连分数形态
(q[1],q[2],q[3],...,q[m])=ContinuedFraction(e,n);
//连分数收敛值的初始条件
c[0]=1;
c[1]=q[1];
d[0]=0;
d[1]=1;
for(j=1;j<=m;j++)
    if((d[j]*e-1)%c[j]==0)
        {
            phi=(d[j]*e-1)/c[j];
            d=d[j];
            b=n-phi+1;
            p=(b-sqrt(b^2-4*n))/2;
            q=n/p;
            return (d,p,q);
        }
    else
        {
        c[j]=q[j]*c[j-1]+c[j-2];
        d[j]=q[j]*d[j-1]+d[j-2];
        };
    return("Failure");
};
```

例 6.18 Eve 截收到 Alice 传递给 Bob 用 RSA 加密的一条密文

$$c = 669801725341794668,$$

并且知道 Bob 的公开密钥为

$$(n, e) = (8025564884382364573, 4021224022722915139),$$

Eve 试着用 Wiener 连分数攻击法攻击，她首先计算 $\frac{e}{n}$ 的连分数表达式的各收敛值 $\frac{c_j}{d_j}$，得

$$0,1,\frac{1}{2},\frac{238}{475},\frac{1191}{2377},\frac{1429}{2852},\frac{2620}{5229},\frac{9289}{18539},\frac{67643}{135002},\cdots$$

她逐项计算 $\frac{d_j e-1}{c_j}$，到第 8 项时发现，该项为整数，因此她判断 Bob 的私钥为 18439，而且

$$\phi(n)=(d_j e-1)/c_j=8025564878594049280,$$

解一元二次多项式

$$x^2+(n-\phi(n)+1)x+n$$

得解

$$q=2302056641$$
$$p=3486258653,$$

既然已破解得密钥值，她想要观看到底明文是什么，计算

$$\begin{aligned}m&=c^d \pmod n\\&=669801725341794668^{18539} \pmod{8025564884382364573}\\&=3503345971441917457\\&=(23,9,5,14,5,18,0,3,18,13,11,19)_{27}\\&=\text{'wiener cracks'}。\end{aligned}$$

在上例中，其实 d 略大于 $\frac{1}{3}\sqrt[4]{n}$，但 Wiener 连分数攻击依然有效，即使是有些密钥值的长度略长于模数 n 的 1/4，也能成功。这其实说明了 Wiener 连分数攻击，在定理 6.6 中所给的条件并不是最佳化的，但在笔尝试着 $\log_n d>0.3$ 时，Wiener 攻击法就无法奏效，另外质数 p、q 的条件改变，所得到的结果也会有所改变。

6.11 Rabin 密码

在此，我们介绍一种类似 RSA 密码的公开密钥密码系统，即 Rabin 密码，其加密解密如下。

算法 6.6（Rabin 密码） Alice 想加密传递明文 m 至 Bob。

- （密钥产生）令 p、q 为大质数且 $p\equiv q\equiv 3 \pmod 4$。p、q 为私钥，Bob 计算 Rabin 模数 $n=pq$，公开它，作为公钥。
- （Alice 加密）密文 $=c\equiv m^2 \pmod n$（m 为明文）。
- （Bob 解密）由费马小定理知 $m^p\equiv m \pmod p$，$m^q\equiv m \pmod q$。

$$\Rightarrow \begin{cases}m^{p+1}\equiv m^2\equiv c \pmod p\\ m^{q+1}\equiv m^2\equiv c \pmod q\end{cases}$$

$$\Rightarrow \begin{cases} m \equiv \pm m^{\frac{p+1}{2}} \equiv \pm c^{\frac{p+1}{4}} \pmod p \\ m \equiv \pm m^{\frac{q+1}{2}} \equiv \pm c^{\frac{q+1}{4}} \pmod q \end{cases}$$

由中国余式定理得

$$m \equiv aq(\pm c^{\frac{p+1}{4}}) + bq(\pm c^{\frac{q+1}{4}}) \pmod n$$

其中 $aq + bq = 1$ （∵ xeuclidean()）。

注 6.8

- 取质数

$$p \equiv q \equiv 3 \pmod 4$$

的好处，是此时解平方根

$$x^2 \equiv c \pmod p$$

只需要计算

$$x \equiv \pm c^{\frac{p+1}{4}} \pmod p\text{。}$$

而当 $p \equiv 1 \pmod 4$ 时，就得尝试用一种 Las Vegas 算法计算。

- 加密函数在 m 与 n 互质时为 $4 \to 1$ 函数。解密时，相同密文，可能有 4 种明文。这在解密时有诸多不便。因此使用 Rabin 密码时，往往再加上若干长度的额外信息，或其他改进方法。在加密的效率上，只需一次模平方计算，该 RSA 密码加密就会快许多；而解密时，是用中国余式定理计算，大约等同中国余式定理修订版 RSA 解密程序的效率。

例 6.19 Alice 欲使用 Rabin 密码加密明文'rabincipher'并传递信息至 Bob，事先约定数字代码：$a = 0$、$b = 1$、…、$z = 25$，并在信息末尾补上 $z = 25$。

- （密钥产生）Bob 取质数

$$p = 468104551\text{、}q = 338523167\text{，}$$

计算 Rabin 模数

$$n = pq = 158464235091633017$$

并公开它。

- （加密）Alice 取得 Rabin 模数 $n = 158464235091633017$，欲加密明文'rabincipher'，首先在信息末尾补上 $z = 25$，并化为数字代码：

$$\begin{aligned} m &= \text{'rabincipherz'} \\ &= (17, 0, 1, 8, 13, 2, 8, 15, 7, 4, 17, 25)_{26} \\ &= 62403061532383467, \end{aligned}$$

计算

$$\begin{aligned} c \equiv m^2 &\equiv 62403061532383467^2 \pmod n \\ &\equiv 110663140006679971 \pmod{158464235091633017} \end{aligned}$$

传递给 Bob。

- （解密）Bob 收到 $c = 110663140006679971$，计算

$$\pm c^{\frac{p+1}{4}} \pmod p = \pm 163254327$$

及

$$\pm c^{\frac{q+1}{4}} \pmod q = \pm 156440072 。$$

由 xeuclidean()得

$$-27490272p + 38013119q = 1，$$

用中国余式定理得

$$\begin{aligned} m &= \pm(-27490272)(p)(163254327) \pm (38013119)(q)(156440072) \\ &= 96061173559249550, \\ &\quad 129434679706613080, \\ &\quad 62403061532383467, \\ &\quad 29029555385019937 。\end{aligned}$$

其中只有 $62403061532383467 \equiv 25 \pmod{26}$，计算

$$\begin{aligned} m &= 62403061532383467 \\ &= (17, 0, 1, 8, 13, 2, 8, 15, 7, 4, 17, 25)_{26} \\ &= \text{'rabincipherz'}, \end{aligned}$$

去除末尾“z”得明文'rabincipher'。

一般相信 RSA 破解与 RSA 模数的质因数分解相关，即

“能因式分解 RSA 模数 ⇒ 能破 RSA 密码”

而破解 Rabin 密码与分解 Rabin 模数，却有以下的逻辑关系：

“能破解 Rabin 密码 ⇔ 能质因数分解 Rabin 模数”

我们将以下面的简例进行描述。

例 6.20　Eve 截收到密文 c，要对其进行攻击。我们在此说明：

“能破解 Rabin 密码 ⇔ 能质因数分解 Rabin 模数 n”

(⇐) Eve 会分解

$$n = 158464235091633017 = 468104551 \times 338523167，$$

她便可如 Bob 般解密。

(⇒) Eve 能够解此模数的任何一个密文，即她会解

$$x^2 \equiv c \pmod{158464235091633017}，$$

得

$$\begin{aligned} x &= 96061173559249550, \\ &\quad 129434679706613080, \\ &\quad 62403061532383467, \\ &\quad 29029555385019937 。\end{aligned}$$

计算各解的差与模数 n 的 gcd：

$$\gcd(129434679706613080 - 96061173559249550, n) = 338523167，$$

$$\gcd(129434679706613080 - 62403061532383467, n) = 468104551，$$

即分解了模数 $n = 158464235091633017 = 468104551 \times 338523167$。

由上例可知，Rabin 密码是不如选择密文攻击的，因为若 Eve 能解密任一密文，则她也能质因数分解 Rabin 模数，这样她就能如 Bob 般解密；这也说明，公开密钥密码系统所用的单向陷门函数，其破解的条件，最好是解某数学难题的“必要条件”，而不是“充分必要条件”。RSA 密码与 Rabin 密码的安全性虽然同样是建立在因数分解的难题上，但由上述原因说明了为何 RSA 密码优于 Rabin 密码。Rabin 密码在使用上，建议配合 Hash 函数使用。另外多数针对 RSA 的密码的攻击法，略加修改也可适用于 Rabin 密码。

第 7 章　非对称密钥密码系统与离散对数

为何使用公开密钥密码？在 AES 开始使用以来，对称密钥密码所能提供的安全度，甚至比一般长度的 RSA 还要强，另一方面，对称密钥密码运算速度快，适用于大批资料加密，然而在密钥分配与密钥管理上，却衍生了相当麻烦的问题。例如：当 Alice 与 Bob 欲使用任何对称密钥密码加密解密，必事先利用安全管道约定所用的密钥；当人数较多时，假设有 n 个人，任意二人均可利用安全管道约定所用的密钥，此时就要 $C_n^2 = \dfrac{n(n-1)}{2}$ 串不同的密钥，在一个中型机构，假设有 5000 人，此时就要有 $C_{5000}^2 = 24995000$ 个不同的密钥，而每人要保存 4999 串不同密钥，这对成本管理有诸多不便。

反观公开密钥密码系统，每人各有一串密钥，即一把公钥及一把私钥，其中公钥公开而私钥保密，任何使用者均可在所公布的公钥目录上，查到所有通信者的公钥；比如 Alice 想加密一段重要信息与 Bob 通信，她在目录上找到 Bob 的公钥，然后用 Bob 的公钥加密信息，将密文传至 Bob，Bob 收到 Alice 传来的密文，再用自己的私钥，就可解密并解读明文，如 RSA 算法。由此便可将密钥数量由原先对称密钥密码的 $\dfrac{n(n-1)}{2}$ 降至 $2n$，而使用者只需要保管好自己的密钥即可。

公开密钥密码的加密、解密函数，一般而言，大都互为反函数，采用计算复杂度较高的单向函数，如模指数运算，其安全性高于短密钥的对称密钥密码，但加密解密速度慢于对称密钥密码。另外一方面，一般公开密钥密码可用私钥进行作用，用来数字签名（Digital Signature），而任何人都可取得其公钥，用以数字签名，通信安全多了一层保障，如此也符合不可否认性，这是对称密钥密码所不具备的。

在计算机以近乎摩尔定律（每 18 个月计算速度加一倍）加速发展的今日，过去硬件所不能负荷的公开密钥密码加密系统，现在已经有了广泛的应用，期间还与对称密钥密码以及其他密码针对不同目的整合成各种加密系统，也成了实际加密计算的典范。

公开密钥密码系统除了 RSA 密码、Rabin 密码外，还有其他许多不同的公开密钥密码系统，这些公开密钥密码系统纷纷建构在不同 NP 复杂性（Nondeterministic Polynomial Completeness）的数学难题上，如质因数分解、离散对数问题（Discrete Logarithm Problem）、迷袋问题（Knapsack Problem）、编码理论（Coding Theory）以及有限自动机（Finite Automation）等问题；有些问题所建立的公开密钥密码系统，安全性实在堪忧，有的也只是纯数学的兴趣，我们在此并不对所有的问题作完整的介绍，因为如此一来，本书将陷于庞杂的不同分歧问题讨论的迷宫中，这不是本书的目的；另一方面，公开密钥密码的发展，逐渐与计算数论（Algorithmic Number Theory）、演算数论（Algorithmic Number Theory）的发展日益相关。因此，在本章中，我们还是将焦点放在那些容易程序化、以模指数运算为主的非对称密码系统上如 Pohlig-Hellman 密码、Diffie-Hellman 密钥交换以及 ElGamal 密码等；至于将来可能取代 RSA 密码系统的椭圆曲线密码系统，因所需数学背景远较上述密码系统多，我们将

在第 10 章给予专门描述。然而这些非对称密钥密码系统，都是与离散对数问题（Discrete Logarithm Problem）有密切关联的。

7.1 Pohlig-Hellman 密码与离散对数

在集合的“包含”概念中，我们知道公开密钥密码系统一定是非对称密钥密码系统，即

$$\{\text{公开密钥密码系统}\} \subset \{\text{非对称密钥密码系统}\}。$$

然而有一种密码系统，它虽是非对称密码系统，但并不是公开密钥密码系统，这就是 Pohlig-Hellman 密码，它属于是私钥密码系统，与 RSA 密码类似，也是以模指数运算为主进行加密、解密。

算法 7.1（Pohlig-Hellman 密码） Alice 与 Bob 协定取一大质数 p。令 m 为明文，c 为密文。则

- （加密）$c \equiv m^e \pmod p$
- （解密）$m \equiv p^d \pmod p$

其中 $ed \equiv 1 \pmod{p-1}$，e 与 d 两值必须保密，否则可用 xeuclidean() 简单推得 $e \equiv d^{-1} \pmod{p-1}$。

假设已经有一对明文及密文，借助对照文攻击，欲求加密密钥

$$e = \log_m c$$

这就是离散对数问题（Discrete Logarithm Problem, DLP）如图 7-1 所示。

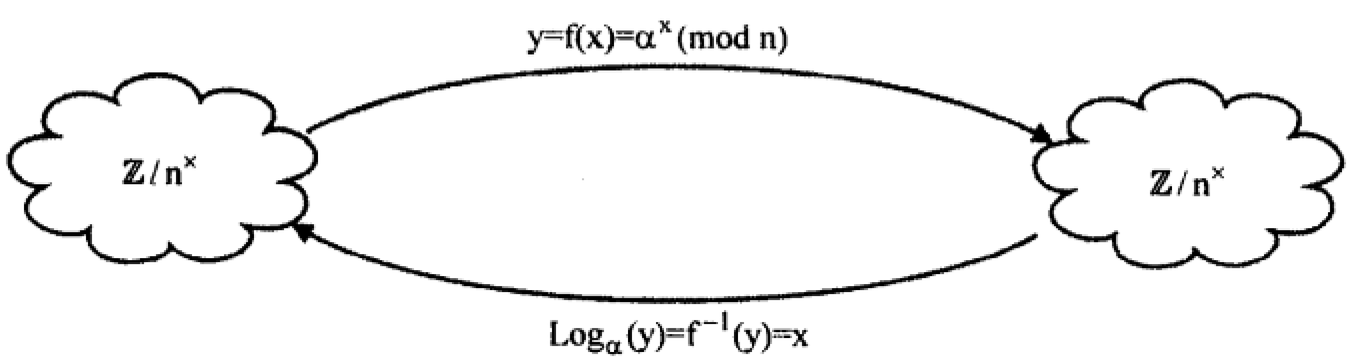


图 7-1 离散对数函数为模指数函数的反函数，模指数函数计算容易，而离散对数函数计算复杂

定义 7.1（离散对数，Discrete Logarithm，Index） 令 n、α、β 为非零整数，满足

$$\beta \equiv \alpha^x \pmod n，（其中 n 不一定是质数）$$

寻找 $x = \log_\alpha \beta$ 的问题即离散对数问题。

例 7.1 令 $p = 13$、$\alpha = 2$ 而 $2^7 \equiv -2 \pmod{13}$，所以 $\log_2 11 \pmod{13} = 7$，另外 7、19、31、43、…、7+12n、…任何一个值也均为其解（费马小定理）。

在离散对数问题中，一般取 α 为 n 的原根（Primitive Root），即在乘法群 $\mathbb{Z}/n^\times$ 中，所有的 β 都可表示为 α 的若干次方，详述如下：

定理 7.1（原根） 令 n、α 为非零整数且 $\gcd(\alpha, n) = 1$，当且仅当以下任一等价关系成立，α 为 n 的原根：

（1）α 生成 $\mathbb{Z}/n^\times$，即

$$<\alpha> := \{\alpha^i \bmod n \mid i \in \mathbb{N}\} = \mathbb{Z}/n^\times，$$

（2）α 的秩（Order）为

$$\mathrm{ord}(\alpha)=\phi(n)=\#\mathbb{Z}/n^{\times},$$

（3）加法群 $(\mathbb{Z}/\phi(n),+)$ 与乘法群 $(\mathbb{Z}/n^{\times},\cdot)$ 同构（Isomorphic），即

$$f:\mathbb{Z}/\phi(n)\to\mathbb{Z}/n^{\times}\quad(x\mapsto f(x)=\alpha^{x})$$

满足 $f(x+y)=f(x)f(y)$，且 f 为 1－1 映射。

注意：本定理为原根的代数描述、群论术语，不熟悉群论的读者对此不要惊奇，可参阅本书的"原根"一节。并非所有的 n 都有原根，事实上 n 有原根只在以下条件才成立：

- $n=p^m$，其中 $m\in\mathbb{N}$、p 为奇质数。
- $n=2p^m$，其中 $m\in\mathbb{N}$、p 为奇质数。
- $n=2$、4。

离散对数其实与一般对数有相似的性质。

定理 7.2　令 r、s、n 为非零整数，而 r、s 为 n 的原根，令 x、y 为与 n 互质的整数，则

（1）$\log_r 1\equiv 0\pmod{\phi(n)}$，

（2）$\log_r(xy)\equiv\log_r(x)+\log_r(y)\pmod{\phi(n)}$，

（3）$\log_r x^k\equiv k\log_r x\pmod{\phi(n)}\quad(k\in\mathbb{N})$，

（4）$\log_s x\equiv\log_r(\log_r s)^{-1}\pmod{\phi(n)}$。

证明：我们只证明性质 1，其余可由 Euler 定理推出。由 Euler 定理知

$$r^{\phi(n)}\equiv 1\pmod{n},$$

而 r 为 n 的原根，所以任何整数 $1\leqslant m\leqslant\phi(n)$ 均

$$r^m\not\equiv 1\pmod{n},$$

故得 $\log_r 1\equiv 0\pmod{\phi(n)}$。

例 7.2

解

$$7^x\equiv 3\pmod{13}。$$

已知

i	1	2	3	4	5	6	7	8	9	10	11	12
2^i mod 13	2	4	8	3	6	12	11	9	5	10	7	1

利用 xeuclidean()计算得

$$11^{-1}\equiv 11\pmod{12},$$

故

$$\begin{aligned}x&=\log_7 3\\&\equiv\log_2 3(\log_2 7)^{-1}\pmod{12}\\&\equiv 4\times 11^{-1}\pmod{12}\\&\equiv 4\times 11\pmod{12}\\&\equiv 8\pmod{12}\end{aligned}$$

以离散对数问题为基础的公开密钥密码系统，往往需要原根，以乘法群 $\mathbb{Z}/p^{\times}$（p 为质数）

为例，由于它与加法群 $\mathbb{Z}/(p-1)$ 群同构（因为 $\phi(p)=p-1$），且 $\mathbb{Z}/(p-1)$ 的生成元，即在 $\{1,\cdots,p-1\}$ 中与 $p-1$ 互质的数，共 $\phi(p-1)$ 个，所以乘法群 $\mathbb{Z}/p^{\times}$ 的原根共有 $\phi(p-1)$ 个。在实用中，奇质数 p 的选取，尽量使 $p-1$ 含有“大”质数因子，一般而言，在 $\{2,\cdots,p-2\}$ 中，随机选取一值且是原根的概率（因 $\pm1 \pmod p$ 均不可能为原根）为

$$\frac{\phi(p-1)}{p-3}。$$

原根的寻找，可以“概率”式方式寻找，并配合以下判别式。

定理 7.3 令 r 为整数，p 为奇质数，r 为乘法群 $\mathbb{Z}/p^{\times}$ 的原根 $\Leftrightarrow$

$$r^{\frac{p-1}{q}} \not\equiv 1 \pmod p$$

其中对 q 为 $p-1$ 的所有质因子均成立。

证明：($\Rightarrow$) 令 r 为乘法群 $\mathbb{Z}/p^{\times}$ 的原根，故

$$r, r^2, r^3, \cdots, r^{p-2}, r^{p-1} \equiv 1 \pmod p$$

为 $\mathbb{Z}/p^{\times}$ 中所有的元素，故

$$r^{\frac{p-1}{q}} \not\equiv 1 \pmod p,$$

当 q 为 $p-1$ 的质因子时均成立。

($\Leftarrow$) 假设 r 不为 $\mathbb{Z}/p^{\times}$ 的原根，要证明存在某 $p-1$ 的质因子，使得

$$r^{\frac{p-1}{q}} \equiv 1 \pmod p。$$

$r \pmod p$ 为 $\mathbb{Z}/p^{\times}$ 的元素之一，故其所生成的子群 $\langle r \rangle = \{r^i \mid i \in \mathbb{Z}\}$ 满足

$$\#\langle r \rangle = \mathrm{ord}(r) \mid \#\mathbb{Z}/p^{\times} = p-1$$

（根据 Lagrange 定理），但因 r 不为原根，所以 $\mathrm{ord}(r) \neq p-1$，故存在某 $p-1$ 的质因子 q 使得

$$\mathrm{ord}(r) \,\Big|\, \frac{p-1}{q}$$

因此

$$r^{\frac{p-1}{q}} \equiv 1 \pmod p$$

例 7.3 以乘法群 $\mathbb{Z}/2^{\times}$ 而言，为秩 $\#\mathbb{Z}/29^{\times} = \phi(29) = 28$ 的循环群，共有

$$\phi(28) = \phi(4)\phi(7) = 2 \times 6 = 12$$

个原根，要判断 $r \in \mathbb{Z}/2^{\times}$ 是否为原根，只需检查

$$r^{14} \not\equiv 1 \pmod{29} \text{ 及 } r^4 \not\equiv 1 \pmod{29}$$

即可。其计算结果如表 7-1 所示。

表 7-1 计算结果

r	2	3	5	7	11	13	17	19	23
$r^4 \pmod{29}$	16	23	16	23	25	25	1	24	20
$r^{14} \pmod{29}$	−1	−1	1	1	−1	1	−1	−1	1

其余元素可由表计算，故得原根为

$$2,3,8,10,11,14,15,18,19,21,24,26\text{。}$$

离散对数问题公认是困难的问题；计算函数 $f(x)=\alpha^x \pmod p$ 是容易的，而求反函数 $\log_\alpha \beta$ 并非易事，因为 $f(x)$ 是一个单向陷门函数。然而一般以离散对数问题为基础的公开密钥密码系统，需要“安全质数” p，其实“安全质数”就是 Germain 质数：

“q、$2q+1$ 同时为质数（$p=2q+1$）”。

故寻找安全质数与原根 (p,g) 的概率式算法为：

算法 7.2　输出 n 位安全质数 p 与原根 g：

```
do{
    q=RandomPrime(2^(n-1),2^n-1);//2^(n-1)<=q<2^n
    p=2q+1;
    }
while(!isPrime(p));
do{
    g=RandomInteger(2,p-2);
    }
while((g^q)%p==1);
```

值得一提的是，RSA 质因数分解挑战 2001 年的纪录是，已能分解 155 位十进制数，而离散对数问题已得到解决的是 120 位的十进制数；而两者所用的方法技巧其实有相当的类似性。

7.2　Diffie-Hellman 密钥交换

第一个公开密钥算法，应属 Diffie-Hellman 密钥交换（Key Exchange），这可追溯到 1976 年。Diffie-Hellman 密钥交换主要用于私钥的产生，而不是用来加密或解密。场景为：Alice 与 Bob 在不安全的连线上，共同使用一种对称密钥密码如 RSA 加密解密，而他们想直接在这连线上约定出共同使用的私钥。

算法 7.3（Diffie-Hellman 密钥交换）　Alice 与 Bob 要约定一个共同私钥，如图 7-2 所示。

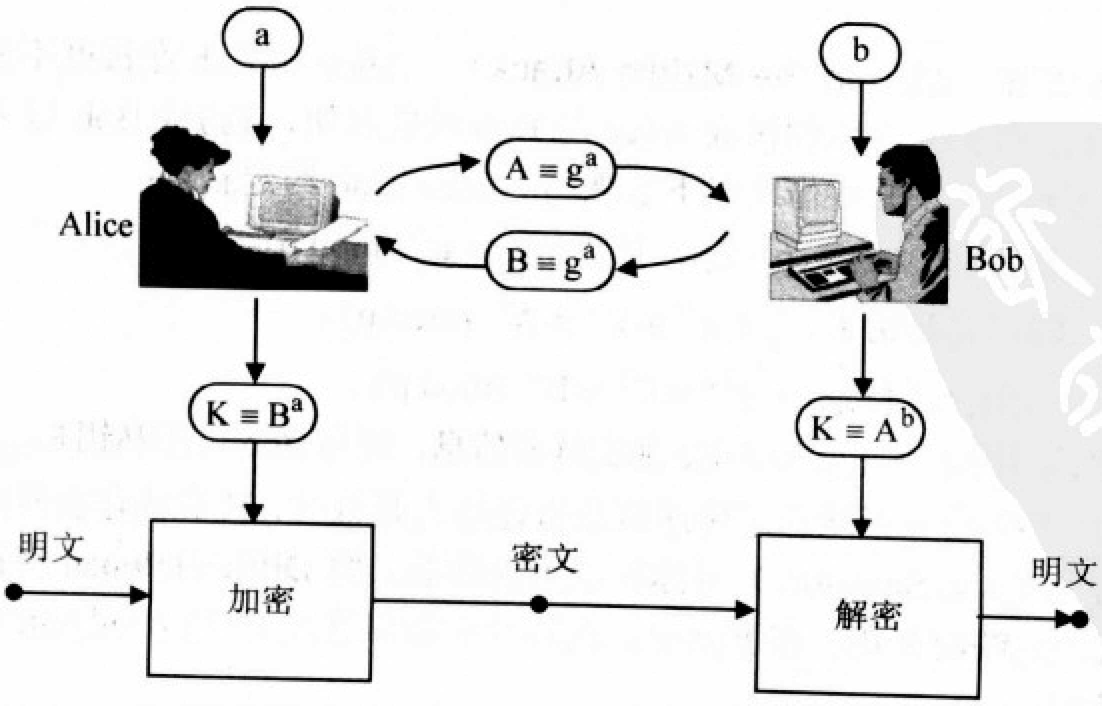


图 7-2　用 Diffie-Hellman 密钥交换约定对称密钥密码密钥

（1）共同取大质数 p 以及底数 g 原根 $(\bmod p)$ $(2 \leqslant g \leqslant p-2)$。

（2）Alice 取 $a \in \{0,1,2,\cdots,p-2\}$，计算 $A \equiv g^a \pmod p$，将值 A 传递给 Bob。

（3）Bob 取 $b \in \{0,1,2,\cdots,p-2\}$ 计算 $B \equiv g^b \pmod p$，将值 B 传递给 Alice。

（4）Alice 收到 B，计算 $K \equiv B^a \equiv (g^b)^a \pmod p$，K 即共同约定的密钥。

（5）Bob 收到 A，计算 $K \equiv A^b \equiv (g^a)^b \pmod p$，K 即共同约定的密钥。

注 7.1 即使是不安全的通道传递信息，Eve 只知道值 p、g、A、B，此时要推算出私钥 $K \equiv g^{ab} \pmod p$ 并非易事，因 $f(x) = g^x \pmod p$ 为单向陷门函数。若值 p、g、A、B 已知，欲求私钥 K，这样的问题称之为 Diffie-Hellman 问题。

此处相关的离散对数问题，即解 $\log_g A$ 或 $\log_g B$，只要她能解出任一离散对数问题，她就知道了 a 或 b 的值，从而可立即计算出私钥值 $K \equiv A^b \equiv B^a \pmod p$。由此可知离散对数问题与 Diffie-Hellman 问题的逻辑关系，即

定理 7.4 能解离散对数问题 ⇒ 能解 Diffie-Hellman 问题。

例 7.4 考虑简例如下：取 $(p,g) = (17,3)$，其中 3 是原根 $(\bmod 17)$。Alice 取 $a = 7$ 计算

$$A \equiv g^a = 3^7 \pmod{17} \equiv 11,$$

并将 $A = 11$ 传给 Bob。Bob 取 $b = 4$，计算

$$B \equiv g^b = 3^4 \pmod{17} \equiv 13,$$

并将 $B = 13$ 传给 Alice。所以共同约定的私钥

$$K = g^{ab} \equiv 3^{7\times 4} \equiv 3^{28} \equiv 4 \pmod{17}.$$

Bob 计算

$$K \equiv A^b \equiv 11^4 \equiv 4 \pmod{17};$$

而 Alice 计算

$$B^a \equiv 13^7 \equiv 4 \pmod{17}.$$

此密钥交换算法在业界广为使用，在许多安全协议中，如 SSL、PKCS 等都有采用。此算法的安全性，是构建在离散对数问题困难度颇高的基础上，但在使用时仍要注意以下的攻击方法。

例 7.5（居中攻击，Man in the Middle Attack） Alice 和 Bob 在极度不安全的通路共同约定私钥，当中第三者 Eve 分别假扮成 Alice 与 Bob 约定私钥，假扮成 Bob 与 Alice 约定私钥，而 Alice 与 Bob 浑然不知。Eve 借助以下步骤与 Alice、Bob 约定私钥：

- Eve 选一数 $c \in \{0,1,2,\cdots,p-2\}$，计算值 $C \equiv g^c \pmod p$。
- 与 Alice 约定出私钥 $K_{Alice} \equiv g^{ac} \equiv C^a \equiv A^c \pmod p$。
- 与 Bob 约定出私钥 $K_{Bob} \equiv g^{bc} \equiv C^b \equiv B^c \pmod p$。

这样，Eve 便可用私钥 K_{Alice} 与 Alice 加密解密信息，而与 Bob 可用私钥 K_{Bob} 加密解密信息。

由此例可知，当接收方不能确定传递信息给他的人是谁时，其实就存在网络安全上的大漏洞，而数字签名（Digital Signature）可弥补以上的漏洞。将 Diffie-Hellman 密钥交换略为修改也可适用于多人共同约定私钥。在下例中，我们说明如何修改 Diffie-Hellman 密钥交换，用于三人共同约定私钥。

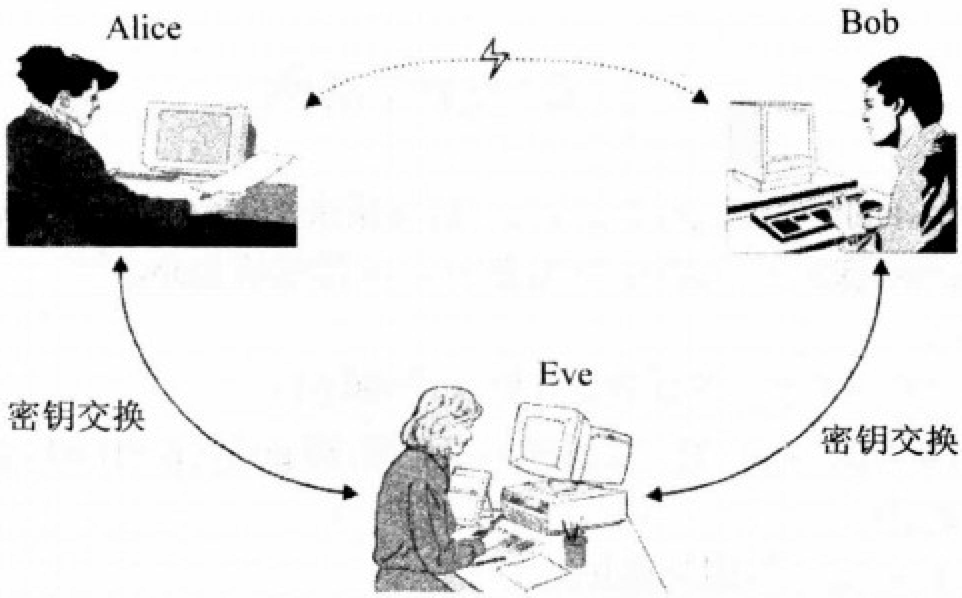


图 7-3　居中攻击

例 7.6　Alice、Bob 与 Carl 欲共同约定私钥，事先约定使用模数 p 为质数，底数 g 为原根 (mod p)。

（1）Alice 取值 a，计算 $A \equiv g^a \pmod p$，将值 A 传递给 Bob。

（2）Bob 取值 b，计算 $B \equiv g^b \pmod p$，将值 B 传递给 Carl。

（3）Carl 取值 c，计算 $C \equiv g^c \pmod p$，将值 C 传递给 Alice。

（4）Alice 计算 $\overline{C} \equiv C^a \pmod p$，传值至 Bob。

（5）Bob 计算 $\overline{A} \equiv A^b \pmod p$，传值至 Carl。

（6）Carl 计算 $\overline{B} \equiv B^c \pmod p$，传值至 Alice。

（7）共同约定的私钥 $K \equiv g^{abc} \pmod p$，而最后计算为：Alice 计算 $K \equiv \overline{B}^a \pmod p$，Bob 计算 $K \equiv \overline{C}^b \pmod p$，而 Carl 计算 $K \equiv \overline{A}^c \pmod p$。

此处 Diffie-Hellman 算法在乘法群 $\mathbb{Z}/p^{\times}$ 上运作，理论上，将 $\mathbb{Z}/p^{\times}$ 代换成任何有限交换群，就可考虑其相对 Diffie-Hellman 算法的推广，其可行与否就要看其间的 Diffie-Hellman 问题或离散对数问题是否够复杂，如椭圆曲线上就有 Diffie-Hellman 算法的推广。而将 Diffie-Hellman 算法适度修改，如 ElGamal 密码，如图 7-4 所示，其实也可用来加密解密。

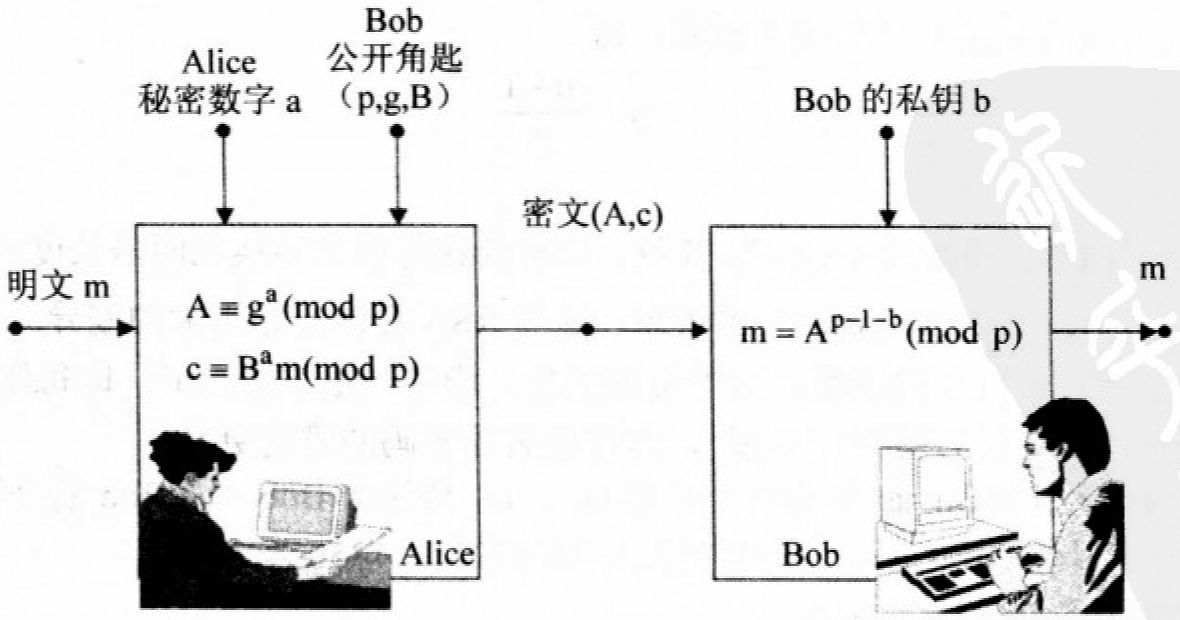


图 7-4　ElGamal 密码系统

7.3 ElGamal 密码

其实只要将 Diffie-Hellman 算法略为修改，即可用来加密解密，即 ElGamal 密码。

算法 7.4（ElGamal 密码） Alice 欲加密明文 m 传递给 Bob。

- （密钥产生）

（1）Bob 选取一大质数 p 以及底数 g 为原根 $\pmod p$。

（2）再取一整数 $b \in \{2,\cdots,p-2\}$，且 b 与 $p-1$ 互质，即 $\gcd(b,p-1)=1$，计算 $B \equiv g^b \pmod p$（同 Diffie-Hellman 算法）。

（3）公开密钥 (p,g,B)，私钥为值 b。

- （加密）明文 $m \in \{2,\cdots,p-1\}$。

（1）Alice 取得 Bob 的公开密钥 (p,g,B)。

（2）随机取整数 $a \in \{0,1,2,\cdots,p-2\}$，计算 $A \equiv g^a \pmod p$（同 Diffie-Hellman 算法）。

（3）计算

$$c = B^a m \pmod p = Km \pmod p$$

（$K \equiv B^a \pmod p$ 为 Diffie-Hellman 密钥交换的密钥）。

（4）密文为 (A,c) 传递给 Bob。

- （解密）Bob 收到密文 (A,c)，令 $x=(p-1-b)$。计算

$$m = A^x c \pmod p = K^{-1} c \pmod p$$

得明文。

证明：

$m = A^x c \pmod p$：

因为 $A^x c \equiv (g^a)^{p-1-b}(B^a m)$

$$\equiv 1 \cdot g^{-ab+ab} m \equiv m \pmod p$$

$$\equiv (g^{p-1})^a ((g^a)^{-b} g^{ab}) m$$

（费马小定理）

ElGamal 密码所用的质数 p 多为安全质数，即

$$p、\frac{p-1}{2}$$

均为质数。

若明文 m 为 l 位，则密文 (A,c) 为 2l 位，加密函数将明文转换成两倍长度的密文，其缺点是有信息扩张（Message Expansion）的现象，这与 RSA 算法有相当大的差异。而 Alice 的数 a 是随机选取的，值 A 为 a 的函数，对于每则信息，均可产生不同的 a'，即每则信息有不同的 Diffie-Hellman 密钥交换的密钥 $K' = B^{a'}$，这可使破译密码的难度更高。

例 7.7 Alice 用 ElGamal 加密两个信息 m_1、m_2 传送给 Bob，而 Bob 的公开密钥为

$$(p,g,B) = (2574233303, 746872111, 135024232)。$$

Eve 截收到这两条信息的密文

$$(A,c_1)=(1971021400,2205685549),$$
$$(A,c_2)=(1971021400,1905838249),$$

并知道 Alice 是用相同的 a 值加密；并知道约定的代码为：空白 = 0、a = 1、b = 2、…、z = 26，且 $[\log_{27}2574233303]=6$，密文每 6 个字母一组编码。但聪明的 Eve 猜到第二条信息应为 'cipher'，故得明文代码

$$m_2='\text{cipher}'=(3,9,16,8,5,18)_{27}$$
$$=48150603;$$

而 $c_2\equiv B^a m_2 \pmod p$，计算 Diffie-Hellman 密钥交换的密钥 $K=B^a$：

$$\begin{aligned}K=B^a &\equiv c_2 m_2^{-1} \pmod p\\ &\equiv c_2 m_2^{p-2} \pmod p\\ &\equiv 1905838249\times 48150603^{2574233301} \pmod{2574233303}\\ &=1295673437。\end{aligned}$$

第一条信息为

$$\begin{aligned}m_1 &\equiv B^a c_1 \pmod p\\ &\equiv 1295673437\times 2205685549 \pmod{2574233303}\\ &=52634826\\ &=(3,18,1,3,11,0)_{27}='\text{crack}',\end{aligned}$$

原来完整信息为 'crack cipher'。

Eve 在未解出乘法群 $\mathbb{Z}/2574233303^{\times}$ 的离散对数问题的情况下破译密文。由此可见，ElGamal 密码若用固定 a 值编码，是抵挡不住明文攻击的。要防止此类攻击，就要利用 ElGamal 密码可作为概率式密码的特性，每条信息均采用不同的 a 进行编码。

7.4　Pohlig-Hellman 算法

计算离散对数的算法中，当循环群 G 的秩仅含有小质因子时，即

$$\#G=\prod_i q_i^{e_i}\quad（q_i 为“小”质数），$$

可利用中国余式定理暨 Pohlig-Hellman 算法计算。以乘法群 $\mathbb{Z}/p^{\times}$ 为例，其秩为 $p-1$，令 α 为该乘法群的一个原根，欲计算 $\alpha^x\equiv\beta \pmod p$，令 q^e 为 $p-1$ 的某质数幂次因子，借由中国剩余定理，可先计算 $x \pmod{q^e}$。先将 x 写成 q 进位数（q-adic）

$$x=x_0+x_1q+x_2q^2+x_3q^3+\cdots+x_iq^i+\cdots\quad(0\leqslant x_i\leqslant q-1)。$$

如何决定系数 x_0、x_1、x_2、…？从同余式

$$\beta\equiv\alpha^x=\alpha^{x_0+x_1q+x_2q^2+\cdots}$$

出发，将式子两边乘上幂次 $\frac{p-1}{q}$，可得

$$\beta^{\frac{p-1}{q}} \equiv \alpha^{x(p-1)/q}$$
$$\equiv \alpha^{(x_0+x_1q+x_2q^2+\cdots)(p-1)/q}$$
$$\equiv \alpha^{x_0(p-1)/q}\alpha^{(p-1)(x_1+x_2q+x_3q^2+\cdots)}$$
$$\equiv \alpha^{x_0(p-1)/q} \pmod p$$

要求得 x_0，只需要逐次计算

$$\alpha^{k(p-1)/q} \pmod p \quad (k=0,1,2,\cdots,q-1)$$

直至某 k 满足 $\alpha^{k(p-1)/q} \equiv \beta^{(p-1)/q}$。即得 $x_0=k$。若 $q^2 \mid p-1$，可利用类似此法继续求 x_1。令

$$\beta_1 \equiv \beta\alpha^{-x_0} \equiv \alpha^{q(x_1+x_2q+x_3q^2+\cdots)} \pmod p 。$$

将同余式两边乘上 $\frac{p-1}{q^2}$ 幂次，可得

$$\beta_1^{\frac{p-1}{q^2}} \equiv \alpha^{(p-1)(x_1+x_2q+\cdots)/q}$$
$$\equiv \alpha^{x_1(p-1)/q}(\alpha^{p-1})^{x_2+x_3q+\cdots}$$
$$\equiv \alpha^{x_1(p-1)/q} \pmod p$$

要求得 x_1，只需要逐次计算

$$\alpha^{k(p-1)/q} \pmod p \quad (k=0,1,2,\cdots,q-1)$$

直至 $\alpha^{k(p-1)/q} \equiv \beta_1{}^{(p-1)/q^2}$ 时，得 $x_1=k$。此方法可逐次进行，解出 x_2， x_3， …， x_{e_1}。

例 7.8 解离散对数问题

$$7^x \equiv 12 \pmod{41} 。$$

- 乘法群 $\mathbf{Z}/41^\times$ 的秩为

$$41-1=2^3\times 5$$

$(q=2)x \equiv x_0+2x_1+4x_2 \pmod 8$。由于

$$\beta^{(p-1)/2} \equiv 12^{20} \equiv 40 \equiv -1 \equiv \alpha^{x_0(p-1)/2}$$

且

$$\alpha^{(p-1)/2} \equiv 7^{20} \equiv -1 \pmod{41},$$

故 $x_0=1$。而

$$\beta_1 \equiv \beta\alpha^{-x_0} \equiv 12\times 7^{-1} \equiv 31 \pmod{41},$$

又

$$\beta_1{}^{(p-1)/2^2} \equiv 31^{10} \equiv 1 \equiv (\alpha^{(p-1)/2})^{x_1} \equiv (-1)^{x_1} \pmod{41},$$

故 $x_1=0$。再者，

$$\beta_2 \equiv \beta_1\alpha^{-2x_1} \equiv 31\times 7^0 \equiv 31 \pmod{41},$$

且

$$\beta_2{}^{(p-1)/q^3} \equiv 31^5 \equiv -1 \equiv (\alpha^{(p-1)/2})^{x_2} \equiv (-1)^{x_2} \pmod{41},$$

所以 $x_2 = 1$，故得

$$x \equiv x_0 + 2x_1 + 4x_2 = 1 + 4 = 5 \pmod 8。$$

- $(q = 5)$ 求 $x \pmod 5$ 如下：计算

$$\beta^{(p-1)/5} \equiv 12^8 \equiv 18 \pmod{41}$$

及

$$\alpha^{(p-1)/q} \equiv 7^8 \equiv 37 \pmod{41},$$

再计算 $\alpha^{k(p-1)/q}$（$k = 0,1,2,3,4$）：

$$37^2 \equiv 16, 37^3 \equiv 18, 37^4 \equiv 10,$$

因此

$$x \equiv 3 \pmod 5。$$

- 由中国余式定理解联立同余式

$$\begin{cases} x \equiv 5 \pmod 8 \\ x \equiv 3 \pmod 5 \end{cases}$$

得

$$x \equiv 5(5 \times 5) + 3(2 \times 8) \pmod{40} = 13,$$

（其中 $5 \times 5 \equiv 1 \pmod 8$，$2 \times 8 \equiv 1 \pmod 5$），故得解 $x = 13$。

7.5 Index Calculus

计算离散对数的算法，最迅速的应为 Index Calculus，在实际操作中通常是结合筛法，如二次筛法或代数数体筛法使用，故其计算复杂度的估计等同于使用相同筛法的因数分解。Index Calculus 适用于乘法群 $\mathbf{Z}/p^{\times}$，但对于椭圆曲线有理点 $E(\mathbb{F}_q)$（其中 $\mathbb{F}_q$ 为 Galois 域）所成的代数群，就几乎没有用武之地。解离散对数问题 $\beta \equiv \alpha^x \pmod p$，其中 p 为大质数而 α 为一原根。

算法 7.5（Index Calculus） 求离散对数问题 $\beta \equiv \alpha^x \pmod p$，其中 p 为大质数而 α 为一原根。

- （选取因数基底 S）如同筛法选取“小”质因数基底

$$S = \{p_1, p_2, \cdots, p_m\}。$$

- （建构同余方程组）对若干随机整数 k $(0 \le k \le p_1)$，计算值 α^k。尝试将 α^k 写成 S 中的元素幂次的乘积，即

$$\alpha^k \equiv \prod_i p_i^{e_i} \pmod p$$

式子两边取离散对数，得

$$k \equiv \sum_i e_i \log_\alpha(p_i) \pmod{p-1},$$

解 $\log_\alpha(p_i)$ $(p_i \in S)$。

- （计算 x）随机取整数 r，计算值 $\beta\alpha^r \pmod p$，使得其值可表为 S 中元素幂次的乘积，即

$$\beta\alpha^r \equiv \prod_i p_i^{d_i} \pmod p,$$

取离散对数可得

$$\log_\alpha(\beta) \equiv -r + \sum_i d_i \log_\alpha(p_i) \pmod{p-1}。$$

例 7.9　求 $2^x \equiv 108 \pmod{269}$ 的解（其中 269 为质数）。

（1）取质因数基底 $S = \{2,3,5,7,11\}$。

（2）有效地计算 $6^k \pmod{269}$：

$$\begin{aligned}
2^{246} &\pmod{269} = 245 = 5\times 7^2\\
2^{113} &\pmod{269} = 48 = 2^4\times 3\\
2^{71} &\pmod{269} = 33 = 3\times 11\\
2^{40} &\pmod{269} = 196 = 2^2\times 7^2\\
2^{109} &\pmod{269} = 3\\
2^{130} &\pmod{269} = 84 = 2^2\times 3\times 7
\end{aligned}$$

取离散对数 $\log_2(\cdot)$ 得

$$\begin{aligned}
246 &\equiv \log_2 5 + 2\log_2 11 \pmod{268}\\
113 &\equiv 4 + \log_2 3 \pmod{268}\\
71 &\equiv \log_2 3 + \log_2 11 \pmod{268}\\
40 &\equiv 2 + 2\log_2 7 \pmod{268}\\
109 &\equiv \log_2 3 \pmod{268}\\
130 &\equiv 2 + \log_2 3 + \log_2 7 \pmod{268}
\end{aligned}$$

$$\Rightarrow \begin{bmatrix} 0 & 0 & 1 & 0 & 2\\ 4 & 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 2 & 0 & 0 & 2 & 0\\ 2 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \log_2 2\\ \log_2 3\\ \log_2 5\\ \log_2 7\\ \log_2 11 \end{bmatrix} \equiv \begin{bmatrix} 246\\ 113\\ 71\\ 40\\ 130 \end{bmatrix} \pmod{268}。$$

（3）解联立同余方程组 $\log_2 3 \equiv 109 \pmod{268}$，$\log_2 5 \equiv 322 \pmod{268}$，$\log_2 7 \equiv 19 \pmod{268}$，$\log_2 11 \equiv -38 \pmod{268}$。

（4）选取整数 $k = 188$，因

$$\beta 2^k = 108\times 2^{188} \pmod{269} = 77 = 7\times 11,$$

取离散对数 $\log_2(\cdot)$ 可得

$$\log_2\beta + 108 \equiv \log_2 7 + \log_2 11 \pmod{268}$$

$$\Rightarrow x = \log_2\beta \equiv (\log_2 7 + 2\log_2 11 - 188) \pmod{268} = 61。$$

Index Calculus 中质因数基底的选取，可用二次筛法或代数数域筛法，其中代数数域筛法在质数大时最有效率，其计算复杂度为

$$\exp((1.923 + o(1))(\log p)^{1/3}(\log\log p)^{2/3})。$$

第 8 章 数字签名

传统的手写签名或盖章，往往代表签名者的责任，是具有法律效力的；在电子文件充斥的今天，一种用以辨识签署人身份的“电子签名”（Electronic Signature）的方法，就显得尤其重要；而能实现电子签名的方法很多，如利用 DNA 辨识、指纹电子辨识等各种不同的方式；其中利用非对称钥密码暨公开密钥密码技术所达成的电子签名，即称为数字签名（Digital Signature）。

数字签名的应用面很广，因具有“不可否认性”的特性，可以达到对称密钥密码所不能达到的功能，它的使用范围已经不仅是密码学最初考虑的加密问题而已。在实际应用中，公开密钥的数字签名技术可用来发展公开密钥基础设施（PKI）以及数字认证。在选举频繁的中国台湾地区，将来，甚至可以考虑“电子投票”（这就要考虑所谓的盲签名）；当然其他的应用也包括电子商务、电子公文交换等。常用的数字签名方法很多，大多数只要有公开密钥加密的算法，经过略为修改得到一个数字签名算法，如与质因数分解问题相关的公开密钥密码系统 RSA 及 Rabin，都有 RSA 数字签名及 Rabin 签名，都是可回复式的数字签名。而以离散数学对数问题为基础的数字签名算法，有 ElGamal 数字签名、DSA、Schnorr 数字签名、Nyberg-Rueppel 数字签名等，这类数字签名演算与 ElGamal 加密算法相近，算是数字签名算法的主流，除了 Nyberg-Rueppel 数字签名之外，其他都是附录式的。然而在许多应用上，需要“可回复式”的数字签名，如数字认证、盲签名。

密码学还有一个重要元件，即 Hash 函数，Hash 函数是一固定长度输出的单向函数，在许多情况下，Hash 函数往往与其他公开密钥数字签名算法配合使用。目前 Hash 函数的主流是 SHA 系列以及欧洲所研制的 RIPEMD-160，这些都与早期的 MD 系列类似，采用压缩函数多次迭代设计，然而因为都是固定输出长度，都可借“生日攻击”将有效安全位长度减半；而“生日攻击”也可适用于解离散对数问题。只要所使用的数字签名法用到 Hash 函数，就必须了解这一些。

8.1 数字签名方案

一般而言，数字签名，是以公开密钥密码系统技术为主，并配合其他密码元件（如 Hash 函数）整合而成。体现的方法有可回复式方案（Recovery Scheme）以及附录式方案（Appendix Scheme）两大不同方案，输出的模式有固定式（Deterministic）与随机式（Randomized）。就以之前所提的 RSA 数字签名而言，就是一种固定式、可回复式数字签名方案。

定义 8.1 数字签名方案（不论是可回复式或是附录式）是随机式的，如输入固定的文件 m 的数字签名 s，会随输入次数的改变而改变；反之，若固定输入的文件 m 的数字签名 s 不随输入次数的改变而改变，每次都是相同的，则称之为固定式的。

在区别可回复式与附录式的数字签名方案时，可直接从其签名过程以及验证过程的算法得出。

算法 8.1（附录式数字签名方案）　Alice 要将文件 m 签署成 s 给 Bob，Bob 验证数字签名 s。

- 数字签名

（1）Alice 用数字签名函数 $S_A(\cdot)$ 对文件 m 进行作用，得

$$s^* = S_A(m)$$

（2）将数字签名

$$s = (m, s^*)$$

传递给 Bob。

- 验证

（1）Bob 取得 Alice 的公钥，即取得一组不同验证函数，如 $V_{A_1}(\cdot)$、$V_{A_2}(\cdot)$。

（2）计算 $V_{A_1}(s)$、$V_{A_2}(s)$。

（3）比较值 $V_{A_1}(s)$、$V_{A_2}(s)$，若皆相等，就验收无误，否则拒收。

注意：在此类数字签名方案中，若使用 Hash 函数 $h(\cdot)$，则在作用数字签名函数时，以 $S_A \circ h(\cdot)$ 取代 $S_A(\cdot)$，如图 8-1 所示。

算法 8.2（可回复式数字签名方案）　Alice 要将文件 m 签署成 s 给 Bob，Bob 验证数字签名 s。

- 数字签名

（1）Alice 用数字签名函数 $S_A(\cdot)$ 作用在文件 m 上，得

$$s = S_A(m)$$

（2）将数字签名传递给 Bob。

- 验证

（1）Bob 取得 Alice 的公钥，即取得验证函数 $V_A(\cdot)$。

（2）Bob 计算 $V_A(s)$。

（3）若该数字签名正确无误，则必 $V_A(s) = m$，可回复原文件。

以 RSA 数字签名为例，如图 8-1 所示，数字签名就是解密函数 $S_A(\cdot) = D_A(\cdot)$，即

$$S_A(m) = m^d \pmod{n};$$

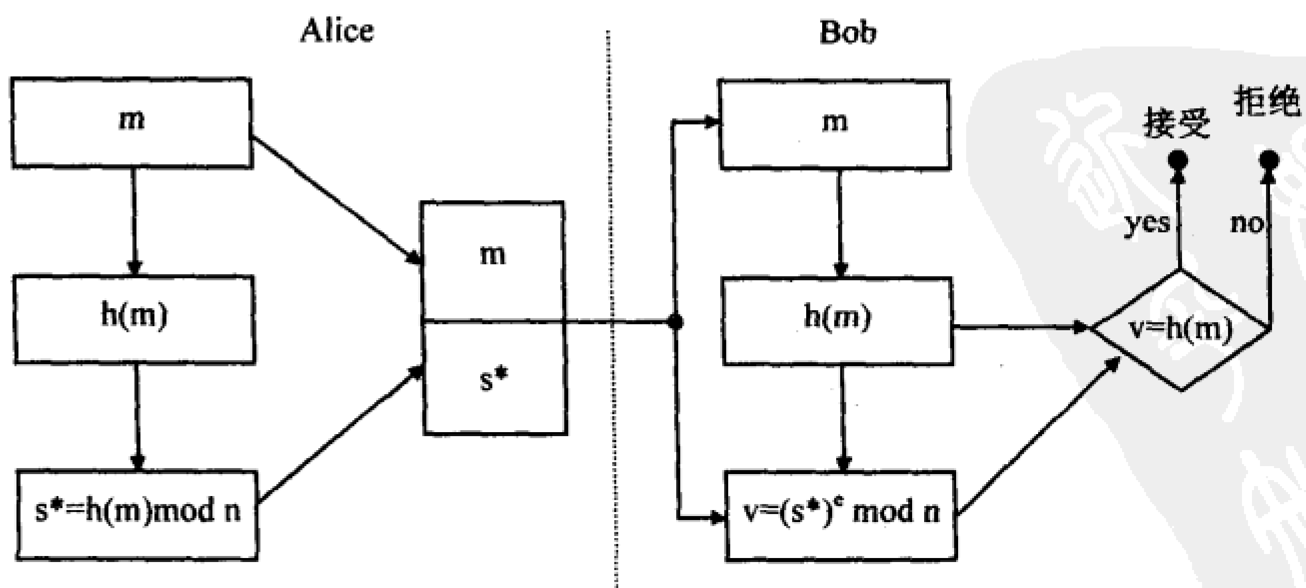


图 8-1　用 Hash 函数的 RSA 数字签名

而验证函数就是加密函数 $V_A(\cdot)=E_A(\cdot)$，即

$$V_A(s)=s^e \pmod n\text{。}$$

例 8.1　我们也可利用 RSA 密码系统以及某 Hash 函数如 SHA-1 整合成附录式数字签名方案。令 RSA 模数 n、加密密钥 e、解密密钥 d 以及 Hash 函数为 $h(\cdot)$。Alice 要将文件 m 数字签名成 s 传送给 Bob，而 Bob 验证数字签名 s。

- （数字签名）Alice 计算

  $$s^*=(h(m))^d \pmod n,$$

  将数字签名

  $$s=(m,s^*)$$

  传递给 Bob。
- （验证）Bob 取得 Alice 的公钥 e，计算

  $$(s^*)^e=((h(m))^d)^e \pmod n$$

  以及

  $$h(m),$$

  比较两值，若相同就验收，否则拒收。

此时数字签名函数为 $S_A(\cdot)=(h(\cdot))^d \pmod n$，而验证函数

$$V_{A_1}(\cdot)=h(\cdot)\text{、}V_{A_2}(\cdot)=(h(\cdot))^e \pmod n\text{。}$$

8.2　RSA 盲签名

单纯的 RSA 数字签名，就是将文件 m 用解密函数进行作用，得数字签名 $s=m^d \pmod n$，验收时只需要以加密函数进行作用，就可回复原文件

$$s^e=(m^d)^e=m^{ed}=m\text{。}$$

而在实际应用中，有时却要求签收者在不知文件内容的情况下，就要得到签收者的数字签名，这特别在电子投票（Electronic Vote）时有其必要性；此时，可回复式的数字签名方案，如 RSA 数字签名，可轻易完成如此的需要，即所谓的盲签名（Blind Signature），其中必须满足两个条件：

（1）签署人对所签署的文件，必须是“盲目”的。

（2）日后公布文件以及数字签名时，签署人也无从找出文件与其盲签名的相关性。

David Chaum 所提的“盲签名”就是利用 RSA 算法。

算法 8.3（RSA 盲签名）　Alice 将文件 m 盲签署成数字签名 s，传递给 Bob，Bob 验证数字签名 s，如图 8-2 所示。

- （密钥产生）Alice 选择 RSA 模数 n（即 $n=pq$，p、q 保密），取加密密钥 e 并公开它，保密解密密钥 d。
- （盲因子）Bob 随机取一整数 k，即为盲因子（Blind Factor），其中 $\gcd(k,n)=1$，计算

  $$t=k^e m \pmod n,$$

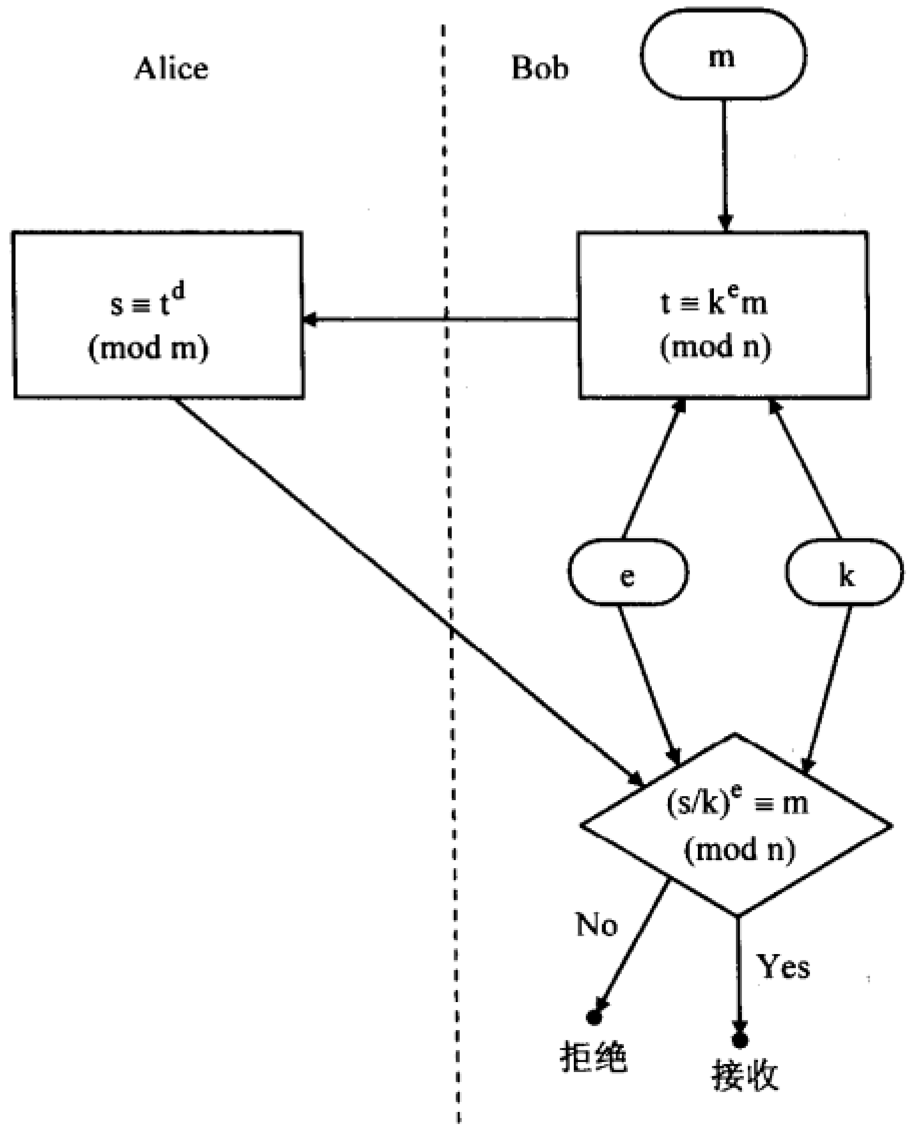


图 8-2 RSA 盲签名

将 t 传送给 Alice。

- （数字签名）Alice 将 t 数字签名成

$$\tilde{s} \equiv t^d \pmod{n},$$

将 $\tilde{s}$ 传回 Bob。

- （去除盲因子）Bob 计算

$$s = \tilde{s}/k \pmod{n}$$

即得 Alice 对 m 的数字签名 $s = m^d$ 。

- （验证）Bob 计算

$$v = s^e = (\tilde{s}/k)^e \pmod{n},$$

若 v = m 则验收，否则拒收。

证明：$\tilde{s}/k = m^d \pmod{n}$：

$$\begin{aligned}\tilde{s}/k \pmod{n} &\equiv t^d/k \pmod{n}\\ &\equiv \frac{(k^e m)^d}{k} \pmod{n}\\ &\equiv \frac{k^{ed} m^d}{k} \pmod{n}\\ &\equiv m^d \pmod{n}\end{aligned}$$

当然在电子投票中，Alice 代表选委会，而 Bob 代表个别的投票人，Bob 当然不希望选委会在他投票后马上得知文件 m 及他到底投谁；而开票后，即便公布了 m 以及 s 之后，选委会 Alice 也无法推算该票是谁投的，同时盲签名的机制也防止选委会 Alice 作票的可能性，这些至少在理论上是可行的。一般而言，凡可回复式的数字签名的方案均可用来盲签名，而这是附录式数字签名方案所不能实现的。

8.3　Hash 函数简介

密码学中有一重要密码元件，即 Hash 函数（中文翻译成杂凑函数或哈希函数），这在数字签名中，特别是与公开密钥密码技术结合整合成附录式的数字签名方案。而 Hash 函数为一个函数，在此，我们用以下定理描述。

定理 8.1（Hash 函数）　Hash 函数 $h(\cdot)$ 为一个函数

$$h:\{任意长度的信息\}\rightarrow\{固定长度的摘要（Digest）\}$$

且满足以下条件：

（1）给定信息 m，计算摘要 $h(m)$ 非常迅速。

（2）给定 y，很难找到 m 使得

$$h(m)=y$$

（此称 $h(\cdot)$ 为单向函数（One-Way Function））。

（3）很难同时找到不同的 m_1 及 m_2 使得

$$h(m_1)=h(m_2)$$

（此称为强抗碰撞（Strongly Collision Resistance）），如图 8-3 所示。

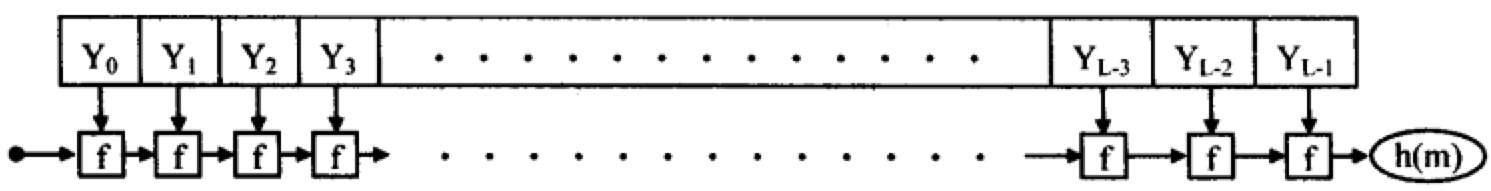


图 8-3　Hash 函数

在数字签名中若使用 Hash 函数，其作法即

$$S\circ h(m)$$

其中 $S(\cdot)$ 为数字签名函数，$h(\cdot)$ 为 Hash 函数，因为摘要 $h(m)$ 是固定长度的，这可大幅加快数字签名的速度。另外，Hash 函数不满足乘法代数性质，即

$$h(m_1m_2)\neq h(m_1)h(m_2),$$

这样可防止利用乘法代数性质的伪造签名攻击；而传送的数字签名为

$$(m,S\circ h(m))。$$

在实际中，常使用的 Hash 函数有 Rivest 所研发的 MD 系列的 Hash 函数以及所谓安全 Hash 算法（即 SHA）和欧洲所研发的 RIPEMD-160，其中 MD 系列多为 128 位，而 SHA、RIPEMD-160 为 160 位；但 128 位的 Hash 函数可用 Oorschot 与 Wiener 所设计造价千万美金的机器在 24 天内找到 *MD5* 的碰撞，128 位的 MD 系列目前已不安全。

一般 Hash 函数包括一个重复使用的压缩函数（Compression Function）f，该压缩函数 f

应有两个输入即两个变量，一个是前次输出，称之为串联变数（Chaining Variable），另一个为文件 m 的一个区块，压缩函数迭代使用，最后的输出即信息摘要 n 位 h(m)。其作法为：

将 m 填补成 m'，即 L 块 b 位区块，

$$m' = Y_0 \| Y_1 \| Y_2 \| \cdots \| Y_{L-1}。$$

然后以压缩函数 f 迭代作用：

$$c_0 = IV = n\text{位初始值}$$
$$c_i = f(c_{i-1}, Y_{i-1}) \quad (1 \leqslant i \leqslant L)$$
$$h(m) = c_L \quad 。$$

压缩函数 f 当然就成为 Hash 函数设计的关键，以抗碰撞的压缩函数 f 设计的 Hash 函数就一定是抗碰撞的。但是依据鸽笼原理（Pigeonhole Principle），只要 $b > 2^n$，就一定会有碰撞，压缩函数 f 就要考虑在计算上很难找到产生碰撞的函数。

8.4 生日攻击

Hash 函数的攻击方式，最常用的就是所谓的生日攻击（Birthday Attack），实质上就是利用概率性质进行随机攻击。

例 8.2（生日攻击） 同一房间内有 23 人，有相同生日的概率为 > 50%，若有 30 人，则相同生日的概率为 > 70%。

为什么？这纯粹是概率问题。假设有 r 个不同信息 m_1，m_2，…，m_r，而在 Hash 函数 $h(\cdot)$ 作用下，有固定长度 l 位，故所有摘要的可能性共有 $n = 2^l$ 个。

r 个信息经 Hash 函数 $h(\cdot)$ 对应到 n 个不同摘要的可能性，共有 n^r 个。而每个摘要最多只有一个信息 m_i 对应到的可能性，共有

$$n(n-1)\cdots(n-r+1)$$

个，故其概率为

$$\begin{aligned} p &= \frac{n^r - n(n-1)\cdots(n-r+1)}{n^r} \\ &= 1 - \frac{n(n-1)\cdots(n-r+1)}{n^r} \\ \Rightarrow 1-p &= \frac{n(n-1)\cdots(n-r+1)}{n^r} \\ &= (1-\frac{1}{n})(1-\frac{2}{n})\ldots(1-\frac{r-1}{n}) \\ &= \prod_{i=1}^{r-1}(1-\frac{i}{n}) \end{aligned}$$

将等式取自然对数 $\ln(\cdot)$，得

$$\ln(1-p) = \sum_{i=1}^{r-1}\ln(1-\frac{i}{n})；$$

而

$$\ln(1-x)=-x-\frac{x^2}{2}-\frac{x^3}{3}-\cdots-\frac{x^k}{k}-\cdots,$$

其中 $|x|<1$，故当 $x\to 0$，$\ln(1-x)\to x$，（其中 $\ln(1-x)<-x$）故可得

$$\ln(1-p)\approx(<)-\sum_{i=1}^{r-1}\frac{i}{n}=-\frac{r(r-1)}{2n}$$

$$\Rightarrow 1-p=e^{\ln(1-p)}\approx(<)e^{-\frac{r(r-1)}{2n}}$$

所以当 $r\approx\sqrt{2\lambda n}$，其有相同摘要的概率

$$p\approx 1-e^{-\lambda}。$$

例 8.3　令某 Hash 函数 $h(\cdot)$ 的摘要为 60 位，故其所有摘要的总数为 $n=2^{60}\approx 10^{18}$，Eve 欲攻击它，并事先找出 35 处可能细微改变，并储存其共 2^{35} 个摘要计算。此时，他只要比较其摘要，就大致可能攻击成功。$r=2^{35}$，$n=2^{60}r=\sqrt{\lambda n}$ 得 $\lambda=2^{10}=1024$，故其攻击成功的概率为 $p\approx 1-e^{-1024}\approx 1$。

由于生日攻击，在使用 Hash 函数时，千万要注意 Hash 函数的摘要的有效安全长度，其实可借助此攻击，将其降低为原有摘要长度的一半，如 MD5 的摘要为 128 位，SHA-1 的摘要为 160 位，其有效安全长度借助生日攻击只剩下 64 位以及 80 位。以 64 位而言，在有效时间的暴力攻击成功已成为可能，因此 Hash 函数的使用，反而降低了公开密钥密码系统本应提供的计算安全度。因此在使用用到 Hash 函数的数字签名的方案时，除了一方面要加长其公开密钥密码系统的密钥长度，另一方面，还要找到摘要“够长”的 Hash 函数，否则，用过短的 Hash 函数，即使所用的公开密钥密码系统或对称密钥密码系统再强，也只是徒然，Hash 函数的使用不可不慎。

生日攻击离散对数

生日攻击也可用于离散对数。欲解离散对数

$$\alpha^x\equiv\beta \pmod p,$$

其中 p 为质数。列出两列计算，均约 $\sqrt{p}$ 长：

（1）第一列计算为 $\alpha^i \pmod p$，其中 i 为随机取的约 $\sqrt{p}$ 值。

（2）第二列计算为 $\beta\alpha^j \pmod p$，其中 j 为随机取的约 $\sqrt{p}$ 值。

两列计算中很有可能有相同者，即

$$\alpha^i\equiv\beta\alpha^j \pmod p \Rightarrow \alpha^{i-j}\equiv\beta \pmod p,$$

故 $i-j \pmod{p-1}$ 为其解。

8.5　ElGamal 数字签名

ElGamal 数字签名为一附录式的数字签名方案，也应该按需要采用随机式数字签名，其算法与 ElGamal 加密法类似，我们将其整理成以下算法。

算法 8.4（ElGamal 数字签名）　Alice 要数字签名一个文件 m 成数字签名 s 传递给 Bob。

- （密钥产生）Alice 选择质数 p 以及其一原根 g，并选择一个整数 a 介于 2、p−2 之间，计算

$$A = g^a \pmod p,$$

将 (p,g,A) 公开，而 a 必须保密。

- 数字签名

（1）Alice 随机取一整数 k 满足 $\gcd(k,p-1)=1$。

（2）计算 $r \equiv g^k \pmod p$。

（3）计算 $s^* \equiv k^{-1}(m-ar) \pmod{p-1}$；将数字签名

$$s=(m,r,s^*)$$

传送给 Bob。

- 验证

（1）Bob 取得 Alice 的公开密钥 (p,g,A)。

（2）计算

$$v_1 \equiv A^r r^{s^*} \pmod p$$

$$v_2 \equiv g^m \pmod p\text{。}$$

（3）检验 $v_1 \equiv v_2 \pmod p$ 是否成立；若成立则验收，否则拒绝。

证明：验证 $v_1 \equiv v_2 \pmod p$ 是否成立，若成立则验收，否则拒绝。因为

$$s^* \equiv k^{-1}(m-ar) \pmod{p-1}$$

$$\Leftrightarrow s^*k \equiv m-ar \pmod{p-1}$$

$$\Leftrightarrow m \equiv s^*k+ar \pmod{p-1}$$

$$\Leftrightarrow v_2 \equiv g^m \equiv g^{s^*k+ar} \equiv v_1 \pmod p\text{。}$$

注 8.1 与 ElGamal 密码一样，所用的质数 p 多为安全质数。若使用 Hash 函数 $h(\cdot)$ 结合 ElGamal 数字签名，所整合的数字签名方案，可将以上算法略为修改如下：

- 数字签名

（1）Alice 随机取一整数 k，使满足 $\gcd(k,p-1)=1$。

（2）计算 $r \equiv g^k \pmod p$。

（3）计算

$$s^* \equiv k^{-1}(h(m)-ar) \pmod{p-1}$$

将数字签名

$$s=(m,r,s^*)$$

传送给 Bob。

- 验证

（1）Bob 取得 Alice 的公开密钥 (p,g,A)。

（2）计算

$$v_1 \equiv A^r r^{s^*} \pmod p$$

$$v_2 \equiv g^{h(m)} \pmod p\text{。}$$

（3）检验 $v_1 \equiv v_2 \pmod p$ 是否成立；若成立则验收，否则拒绝。

注意，在此情况下，Hash 函数 $h(\cdot)$ 是双方事先约定的，由于 ElGamal 数字签名是附录式的，往往与 Hash 函数配合使用。

例 8.4　Alice 用 ElGamal 数字签名信息传送给 Bob，而 Alice 的公开密钥为

$$(p,g,A)=(143988026487479, 11, 92403045857968)。$$

Eve 截收到此信息的数字签名，其中两条

$$\begin{aligned} s_1 &= (m_1,r,s_1^*) \\ &= (\text{'hack bad'}, 89307273047373, 86973759877688), \\ s_2 &= (m_2,r,s_2^*) \\ &= (\text{'signature'}, 89307273047373, 55673729196594), \end{aligned}$$

知道 Alice 是用相同 r 值数字签名(即相同 k 值)；并知道约定代码为：空白 = 0、a = 1、b = 2、…、z = 26，故明文代码为

$$m_1 = (8,1,3,11,0,2,1,4,0)_{27} = 2271216784698,$$

$$m_1 = (19,9,7,14,1,20,21,18,5)_{27} = 5463218140988。$$

聪明的 Eve 计算

$$\begin{aligned} m_1 - m_2 &= -3192001356290 \\ &\equiv (s_1 - s_2)k = 31300030681094k \pmod{p-1}, \end{aligned}$$

由于

$$\begin{aligned} &\gcd(-3192001356290, p-1) \\ =\ &\gcd(-3192001356290, 143988026487479-1) = 2, \end{aligned}$$

将同余式除以 2，得

$$-1596000678145 \equiv 15650015340547k \quad (\text{mod}\ \frac{p-1}{2} = 71994013243739),$$

得 $k \equiv 70166074338643 \quad (\text{mod}\ \frac{p-1}{2})$，故

$$k = 70166074338643 \text{ 或 } 70166074338643 + \frac{p-1}{2} = 142160087582382,$$

但

$$11^{70166074338643} = 89307273047373 \pmod p = r,$$

$$11^{142160087582382} = 76479075265669 \pmod p \neq r,$$

所以可以决定值

$$k = 70166074338643。$$

将已知值代入

$$s_1^* \equiv k \equiv m_1 - ar \pmod{p-1},$$

即

$$\begin{aligned} 70166074338643a \equiv\ & 2271216784698 \\ & - 86973759877688 \times 86973759877688 \\ & (\text{mod}\ 143988026487478), \end{aligned}$$

解得

$$a = 98515169462148,$$

从此 Eve 已经知道 Alice 的密钥 a，可任意伪造 Alice 的 ElGamal 数字签名。

ElGamal 与 RSA 数字签名一样，也有代数乘法性质（因两者的基本运算皆为指数模运算），使用 Hash 函数正可防止利用代数乘法性质的伪造攻击。

例 8.5 计伪造 Alice 的 ElGamal 数字签名 (m,r,s^*)。她取整数 u 以及 v，其中 $\gcd(v,p-1)=1$，伪造

$$r \equiv A^v g^u \pmod p$$
$$s^* \equiv -rv^{-1} \pmod{p-1}$$
$$m \equiv s^* u \pmod{p-1}$$

其中 $m \equiv s^* u \pmod{p-1}$ 居然是有意义的文字的代码（虽然这在 ElGamal 数字签名系统中的可能性微乎其微）。

证明：根据 s^* 的原始定义，即检查

$$s^* k + ar \equiv m \pmod{p-1}$$

是否代入伪造值也成立；此时

$$s^* k + ar \equiv r(-v^{-1}k + a) \pmod{p-1}$$
$$m \equiv s^* u \equiv -rv^{-1}u \pmod{p-1},$$

但是

$$r \equiv g^k \equiv A^v g^u \equiv g^{av} g^u \equiv g^{av+u} \pmod p$$
$$\Leftrightarrow k \equiv u + av \pmod{p-1}$$
$$\Leftrightarrow k - av \equiv u \pmod{p-1}$$
$$\Leftrightarrow v^{-1}k - a \equiv -v^{-1}u \pmod{p-1}$$
$$\Leftrightarrow r(-v^{-1}k + a) \equiv -rv^{-1}u \pmod{p-1},$$

故 $s^* k + ar \equiv m \pmod{p-1}$ 成立。

8.6 DSA 数字签名

美国国家标准技术局（National Institute of Standard and Technology，NIST）在 1991 年提议了一种数字签名算法（Digital Signature Algorithm，DSA），并在 1994 年将其制定为数字签名的标准（Digital Signature Standard，DSS），DSA 其实是 ElGamal 数字签名的变形，其算法大体类似，不同点只是将原根 g 改为另一非原根的值，其算法如下：

算法 8.5（DSA 算法） Alice 要将 m 数字签名成数字签名 s 传递给 Bob，而 Bob 验证 Alice 的数字签名。

- 密钥产生

（1）Alice 选择 160 位大小的质数 q 并选择另一质数 p，使得 $q \mid p-1$（这样的选取，可保证 $\pmod p$ 离散对数够难，p 最初建议为 512 位，后为 1024 位）。

（2）选取 g 为 (mod p) 的原根，并计算 $\alpha \equiv g^{\frac{p-1}{q}} \pmod q$（当中 $\alpha^q \equiv 1 \pmod p$）。

（3）Alice 选择一个整数 a 使得 $1 \leqslant a \leqslant q-1$，并计算 $\beta \equiv \alpha^a \pmod p$。

（4）Alice 将 (p,q,α,β) 公开，而 a 必须保密。

- 数字签名

（1）Alice 随机取一个整数 k，其中 $1 < k < q-1$。

（2）计算 $r \equiv (\alpha^k \pmod p) \pmod q$。

（3）若不用 Hash 函数 $h(\cdot)$，计算

$$s^* \equiv k^{-1}(m + ar) \pmod q,$$

若使用 Hash 函数 $h(\cdot)$，计算

$$s^* \equiv k^{-1}(h(m) + ar) \pmod q$$

（4）将数字签名

$$s = (m, r, s^*)$$

传送给 Bob。

- 验证

（1）Bob 取得 Alice 的公开密钥 (p,q,α,β)。

（2）计算

$$u_1 \equiv s^{*-1} m \pmod q \quad （或 u_1 \equiv s^{*-1} h(m) \pmod q）$$

$$u_2 \equiv s^{*-1} r \pmod q$$

（3）计算 $v = (\alpha^{u_1}\beta^{u_2} \bmod p) \pmod q$。

（4）若 $v = r$ 则验收，否则拒绝。

证明：在此我们证明为何如此验证成立。以不用 Hash 函数 $h(\cdot)$ 为例，从 s^* 的定义得

$$m \equiv -ar + ks^* \pmod q$$

$$\Leftrightarrow s^{*-1} m \equiv -ars^{*-1} + k \pmod q$$

$$\Rightarrow \quad k \equiv s^{*-1} m + ars^{*-1} \pmod q$$

$$\equiv u_1 + au_2 \pmod q.$$

所以

$$\alpha^k = \alpha^{u_1 + au_2} = (\alpha^{u_1}\beta^{u_2} \bmod p) \pmod q,$$

因此 $v = r$。

注 8.2（概率上的错误）　在密钥产生时，s^* 有可能会使得 $s^* \equiv 0 \pmod q$，此时当然不存在 $s^{*-1} \pmod q$，那么必须重新选取 k，但出现这种情况的概率为 2^{-160}，是微乎其微的。一般而言，当签名者发现 $s^* = 0$ 或者 $r = 0$ 时就要重新选取 k。

例 8.6（DSA 简例）　Alice 欲将 m = 'dsa sign' 数字签名成电子签名 s 传递给 Bob，Bob 验证 Alice 的电子签名。

- 密钥产生

（1）Alice 选择质数

$$q = 388093586203,$$

并检验 $kq+1\ (k=2,3,\cdots)$ 是否为质数，当 $k=10$，得

$$p=10q+1=3880935862031$$

为质数。

（2）取 (mod p) 的原根 $g=7$。

（3）计算

$$\alpha=g^{\frac{p-1}{q}}\pmod p=282475249。$$

（4）选择一整数 $a=234621371271$，并计算

$$\beta=\alpha^{a}\pmod p=392622649766。$$

（5）Alice 的公钥为

$$\begin{aligned}&(p,g,\alpha,\beta)\\&=(3880935862031,388093586203,282475249,\\&\ 392622649766)\end{aligned}$$

而值 $a=234621371271$ 保密。

- 数字签名

（1）Alice 随机选取一整数

$$k=253098774209,$$

计算

$$r=\alpha^{k}\ \mathrm{mod}\ p\pmod q=186612001224。$$

（2）约定代码为：空白 = 0、a = 1、b = 2、…、z = 26，故明文代码为

$$\begin{aligned}m&='\text{dsa sign}'\\&=(4,19,1,0,19,9,7,14)_{27}\\&=49217131751。\end{aligned}$$

（3）计算

$$s^{*}=k^{-1}(m+ar)\pmod q=196549201306。$$

（4）将数字签名

$$\begin{aligned}s&=(m,r,s^{*})\\&=('\text{dsa sign}',186612001224,196549201306)\end{aligned}$$

传递给 Bob。

- 验证

（1）Bob 取得 Alice 的公钥 (p,q,α,β) 以及数字签名 $s=(m,r,s^{*})$。

（2）计算代码 $m=49217131751$。

（3）计算

$$u_1=s^{-1}m\pmod q=625514134,$$

$$u_2=s^{-1}r\pmod q=168201244864,$$

$$\begin{aligned}v&=(\alpha^{u_1}\beta^{u_2}\ \mathrm{mod}\ p)\pmod q\\&=186612001224。\end{aligned}$$

（4）因 $r=v$，验收。

DSA 数字签名的特色为：

（1）与 ElGamal 数字签名一样，k 值可每次选取随机信息，成为随机式的数字签名方案。

（2）DSA 质数 p、q 的选取与 ElGamal 密码或数字签名取安全质数虽有不同，但都可使得乘法群 $\mathbf{Z}/p^{\times}$ 上的离散对数问题足够困难。

（3）使用 α（$\alpha^q \equiv 1 \pmod p$）而不用原根 g，是与 ElGamal 数字签名最大的差异；也因为如此，不像 ElGamal 数字签名，若有两条以上数字签名 r 值相同，便可让攻击者 Eve 推算出 k 值和密钥值 a，从而任意伪造 ElGamal 数字签名；在 DSA 中，两个相同的 r 值，只能保证 $k_1 \equiv k_2 \pmod q$，而不同值 $k \pmod p$ 有相同 $\pmod q$ 的值共约有

$$2^{512-160}=2^{342}\ （或\ 2^{1024-160}=2^{864}）$$

种可能性，这样攻击 ElGamal 数字签名的这类方法便对 DSA 无效。

（4）另外在验证时，最费时的指数模运算只需要 3 次，要比 ElGamal 数字签名快许多。

DSA 虽是所谓的数字签名标准 DSS，但在 20 世纪 90 年代，无论学术界、业界均有不同意见，不少厂商如 IBM、Apple、Novell、Microsoft、DEC、Sun 均已在 RSA 算法上投入不少金钱，谁也不愿因 DSA 而遭受损失，反对声浪高涨，对 DSA 的批评，我们整理如下：

（1）DSA 仅能用来作数字签名，不能用来加密，也不能作密钥交换。

（2）DSA 是由美国国家安全局 NSA 独自研发出来，研发过程不透明，可能会暗藏陷门。

（3）DSA 验证过程要比 RSA 验证过程慢 10～40 倍。

（4）RSA 是事实上（de facto）的标准，何必另订一数字签名标准。

（5）选择 DSA 的过程不公开，且没有足够的时间分析其算法。

（6）DSA 可能涉及专利问题，如 Schnorr 数字签名法。

（7）DSA 的密钥实在太短；最早的标准是 512 位，这实在太危险，虽然加长至 1024 位，长期看来，也是不够的。要注意的是，DSA 的安全性是基于离散对数问题的，与 RSA 基于因数分解虽有不同，但只要利用 Calculus Index 攻击，不论是二次筛法或是代数数域筛法，都会造成威胁。

然而 DSA 由于其所产生的数字签名的附录 (r,s^*) 不过才 320 位，这在 IC 卡的运用中有很大的实用价值，另外密钥的产生速度很快。在后来 DSS 的制定中，Hash 函数也以 SHA-1 为标准，而 DSA 的广为使用也逐渐成为了事实。

8.7　Schnorr 数字签名

与 ElGamal 数字签名、DSA 数字签名相同，Schnorr 数字签名也是建立在离散对数问题上的数字签名，也属于附录式数字签名方案，此三种数字签名很类似，尤其是 Schnorr 数字签名与 DSA 相同度颇高，其中 Schnorr 数字签名的算法如下：

算法 8.6（Schnorr 数字签名）　Alice 要将 m 数字签名成数字签名 s 传送给 Bob，而 Bob 验证 Alice 的数字签名。

- （密钥产生）与 DSA 完全相同。

（1）Alice 选择质数 q 并选择另一质数 p，使得 $q \mid p-1$。

（2）选取g为 (mod p)的原根，并计算$\alpha \equiv g^{\frac{p-1}{q}} \pmod{q}$（当中$\alpha^q \equiv 1 \pmod{p}$）。

（3）Alice 选择一整数 a 使得$1 \leqslant a \leqslant q-1$，并计算$\beta \equiv \alpha^a \pmod{p}$。

4．Alice 将(p,q,α,β)公开，而 a 必须保密。

● （数字签名）

（1）Alice 随机取一整数 k，其中$1<k<q-1$。

（2）计算$r \equiv \alpha^k \pmod{p}$以及$e = h(m \| r)$，其中$h(\cdot)$为 Hash 函数，而$m \| r$表示m与r的字串联结，并计算

$$s^* = ae + k \mod q\text{。}$$

（3）将数字签名

$$s = (m, s^*, e)$$

传送给 Bob。

● 验证

（1）Bob 取得 Alice 的公开密钥(p,q,α,β)。

（2）计算

$$v = \alpha^{s^*}\beta^{-e} \mod p$$
$$e' = h(m \| v)\text{。}$$

（3）若$e' = e$则验收，否则拒绝。

由此算法，可观察得到 Schnorr 数字签名与 DSA 的不同之处仅仅在于 Hash 函数$h(\cdot)$的使用方式稍有不同，DSA 直接使用文件的信息摘要$h(m)$，而 Schnorr 数字签名则是使用$h(m \| r)$，其雷同性颇高。Schnorr 数字签名的推出日期早于 DSA，专利期到 2008 年才结束，而且在多国注册专利，为国际性的。虽然美国法院未针对 DSA 作出不利的判决，但对于主张保障专利不受侵权的美国国家技术标准局，在 DSA 的制定上，不仅饱受非议，而且面对诸多的批评，实属难堪。

8.8 Nyberg-Rueppel 数字签名

建立在离散对数问题上的数字签名如 ElGamal 数字签名、DSA 数字签名、Schnorr 数字签名都属于附录式数字签名方案，而 Nyberg-Rueppel 数字签名虽也是建立在离散对数问题的数字签名，却是可回复式的，如此一来略加修改，也可用来盲签名。其他建立在离散对数问题上的数字签名，大都与 Hash 函数配合使用，在此却需要与 Hash 函数性质不同的函数，即重复函数（Redundancy Function）配合使用。

定理 8.2（重复函数，Redundancy Function） 令函数$\tilde{m} = R(m)$为 1－1，若很容易从$\tilde{m}$回复至少$m = R^{-1}(\tilde{m})$，则称 R 为重复函数（Redundancy Function）。

例 8.7 函数

$$R_1 : (\mathbb{Z}/k)^1 \to (\mathbb{Z}/k)^{21}$$
$$m \mapsto m \| m = mk^1 + m$$

以及

$$R_2:(\mathbb{Z}/k)^l \to (\mathbb{Z}/k)^{l+2}$$
$$m \mapsto m\|k-1\|k-1 = mk^2+(k-1)k+k-1$$

均是 Redundancy 函数。

算法 8.7（Nyberg-Rueppel 数字签名）　Alice 欲将 m 数字签名成数字签名 s 传送给 Bob，而 Bob 验证 Alice 的数字签名。

- （密钥产生）与 DSA 完全相同。

（1）Alice 选择质数 q 并选择另一质数 p，使得 $q \mid p-1$。

（2）选取 g 为 $\pmod p$ 的原根，并计算 $\alpha \equiv g^{\frac{p-1}{q}} \pmod q$（当中 $\alpha^q \equiv 1 \pmod p$）。

（3）Alice 选择一整数 a 使得 $1 \leqslant a \leqslant q-1$，并计算 $\beta \equiv \alpha^a \pmod p$。

（4）Alice 将 (p,q,α,β) 公开，而 a 必须保密。

- 数字签名

（1）Alice 计算 $\tilde{m} = R(m)$，其中 $\tilde{m} < p$。

（2）Alice 随机选择一整数 k，其中 $1 \leqslant k \leqslant q-1$。

（3）计算

$$r = \alpha^{-k} \pmod p,$$
$$e = \tilde{m}r \pmod p,$$
$$s^* = ae + k \mod q。$$

（4）将数字签名

$$s = (s^*, e)$$

传送给 Bob。

- 验证

（1）Bob 取得 Alice 的公开密钥 (p,q,α,β)。

（2）计算

$$v \equiv \alpha^{s^*}\beta^{-e} \mod p,$$
$$m' \equiv ve \pmod p。$$

（3）若 m' 落入 R 的值域则验收，即 $m' = \tilde{m} = R(m)$，否则拒绝。

（4）回复 $m = R^{-1}(\tilde{m})$。

证明：欲证 $m' \equiv \tilde{m} \pmod p$：

$$因为\ v \equiv \alpha^{s^*}\beta^{-e} \mod p$$
$$\equiv \alpha^{s^*-ae} \equiv \alpha^s \pmod p$$
$$所以\ m' \equiv ve \pmod p$$
$$\equiv \alpha^k \tilde{m} \alpha^{-k}$$
$$\equiv \tilde{m} \pmod p$$

例 8.8（简易 Nyberg-Rueppel 数字签名）　Alice 要将文件 m = 'rueppel' 数字签名成数字签名 s 传送给 Bob，而 Bob 验证 Alice 的数字签名。

- 密钥产生

（1）Alice 选择质数

$$q = 1116298272183734531 09,$$

并检验 $kq+1\ (k=2,3,\cdots)$ 是否为质数，当 $k=19$，得

$$p = 19q + 1 = 20093368899307221559 63$$

为质数。

（2）选取 $g=2$ 为 $(\bmod p)$ 的原根，并计算

$$\alpha \equiv g^{\frac{p-1}{q}} \pmod p = 262144。$$

（3）Alice 选择一整数

$$a = 72411679331808024518,$$

并计算

$$\beta \equiv \alpha^a \pmod p = 1154066146191019681786。$$

（4）Alice 将 (p,q,α,β) 公开，而 a 必须保密。

- 数字签名

（1）约定代码为：空白 = 0、a = 1、b = 2、…、z = 26，故明文代码为

$$\begin{aligned} m &= \text{'rueppel'} \\ &= (18,21,5,16,16,5,12)_{27} \\ &= 7277879793。\end{aligned}$$

（2）计算

$$\begin{aligned} \tilde{m} &= R(m) = m \| m \\ &= 27^7 m + m \\ &= 76129193211034406772。\end{aligned}$$

（3）随机选择一个整数

$$k = 812148046465791365 15。$$

（4）计算

$$\begin{aligned} r &= \alpha^{-k} \pmod p \\ &= 479806280567735355347 \\ e &= \tilde{m} r \pmod p \\ &= 458309456931670200754 \\ s^* &= ae + k \pmod p \\ &= 772713318690123613 50。\end{aligned}$$

（5）将数字签名

$$\begin{aligned} s &= (s^*, e) \\ &= (77271331869012361350, \\ &\quad 458309456931670200754) \end{aligned}$$

传送给 Bob。

- 验证

（1）Bob 取得 Alice 的公开密钥 (p,q,α,β) 以及数字签名 $s=(s^*,e)$。

（2）计算

$$v \equiv \alpha^{s^*}\beta^{-e} \mod p$$
$$\equiv 980258208214l7237576,$$
$$m' \equiv ve \pmod p$$
$$= 761291932110344 06772$$
$$= (18,21,5,16,16,5,12,18,21,5,16,16,5,12)_{27}$$
$$= \text{'rueppelrueppel'}。$$

（3）回复

$$m = R^{-1}(\tilde{m}) = \text{'rueppel'}。$$

8.9　MD5 Hash 函数

MD5 Hash 函数是 RSA 原创者 R（Rivest）所研发的，另外 MD2、MD4 也是由 Rivest 设计，均为输出为 128 位信息摘要的 Hash 函数，这些 Hash 函数的算法均有类似之处，在此仅介绍 MD5。MD5 一开始是将输入的文件以每 512 位为一区块（Block），而每一区块再分为 16 个 32 位的子区块（Subblock），将其运算，最后输出 4 个 32 位的子区块，并得 128 位的信息摘要，其算法如下。

算法 8.8（MD5 算法）　MD5 Hash 函数 $h(\cdot)$ 将 l 位的数据 m 对应至 128 位的信息摘要 $A\|B\|C\|D$，其中 A、B、C、D 为 32 位串联变量（Chaining Variable）。

- （数据初始处理）添加位，使文件的长度为 $448(\bmod 512)$（若数据的长度已为 $448(\bmod 512)$，则再添加 512 位），添加位首位为 1，其余皆为 0。再将数据 m 的长度附加在后，占 64 位（若 m 的长度很长，$l > 2^{64}$，则取 $l \pmod{2^{64}}$）。因此得字串

$$m' = m\|1000\cdots0000\|l \mod 2^{64}。$$

将 m′ 以每 512 位为一区块，故

$$m' = Y[0]Y[1]Y[2]\cdots Y[L-1],$$

其中 $Y[i]\,(1 \leqslant i \leqslant l-1)$ 为 512 位 $\mathbb{F}_2$ 字串。而每一区块 Y[i] 再细分为 16 个 32 位的子区块，即

$$M[0],\ M[1],\ \cdots,\ M[15],$$

其中 $M[k]\,(l \le k \le 15)$ 为 32 位的 $\mathbb{F}_2$ 字串。

- （函数暨常数的定义）使用 4 个非线形函数

$$F(X,Y,Z) := (X \wedge Y) \vee ((\neg X) \wedge Z)$$
$$G(X,Y,Z) := (X \wedge Y) \vee (Y \wedge (\neg Z))$$
$$F(X,Y,Z) := X \oplus Y \oplus Z$$
$$I(X,Y,Z) := Y \oplus (X \vee (\neg Z))$$

其中 ⊕ 表示 XOR、∧ 表示 AND、∨ 表示 OR，¬ 表示 NOT 逻辑运算。4 个 32 位串联变数的初始值为：

$$A = 0x01234567$$
$$B = 0x89abcdef$$
$$C = 0xfedcba98$$

$$D = 0x76543210$$

定义常数 y[j] 为

$$y[j] := [\, 2^{32} \mid \sin(j+1) \mid \,] \quad (j = 0,1,\cdots,63)\,,$$

详见表 8-1 定义。

表 8-1 $y[j] = [2^{32} \mid \sin(j+1) \mid]$

y[0] = 0xd76aa478	y[1] = 0xe8c7b756	y[2] = 0x242070db	y[3] = 0xc1bdceee
y[4] = 0xf57c0faf	y[5] = 0x4787c62a	y[6] = 0xa8304613	y[7] = 0xfd469501
y[8] = 0x698098d5	y[9] = 0x8b44f7af	y[10] = 0xffff5bb1	y[11] = 0x895cd78e
y[12] = 0x6b901122	y[13] = 0xfd987193	y[14] = 0xa679438e	y[15] = 0x49b40821
y[16] = 0xf61e2562	y[17] = 0xc040b340	y[18] = 0x265e5a51	y[19] = 0xe9b6c7aa
y[20] = 0xd62f105d	y[21] = 0x02441453	y[22] = 0xd8a1e681	y[23] = 0xe7d3fbc8
y[24] = 0x21e1cde6	y[25] = 0xc33707d6	y[26] = 0xf4d50d87	y[27] = 0x455a14ed
y[28] = 0xa9e3e905	y[29] = 0xfcefa3f8	y[30] = 0x676f02d9	y[31] = 0x8d2a4c8a
y[32] = 0xfffa3942	y[33] = 0x8771f681	y[34] = 0x699d6122	y[35] = 0xfde5380c
y[36] = 0xa4beea44	y[37] = 0x4bdecfa9	y[38] = 0xf6bb4b60	y[39] = 0xbebfbc70
y[40] = 0x289b7ec6	y[41] = 0xeaa127fa	y[42] = 0xd4ef3085	y[43] = 0x04881d05
y[44] = 0xd9d4d039	y[45] = 0xe6db99e5	y[46] = 0x1fa27cf8	y[47] = 0xc4ac5665
y[48] = 0xf4292244	y[49] = 0x432aff97	y[50] = 0xab9423a7	y[51] = 0xfc93a039
y[52] = 0x655b59c3	y[53] = 0x8f0ccc92	y[54] = 0xffeff47d	y[55] = 0x85845dd1
y[56] = 0x6fa87e4f	y[57] = 0xfe2ce6e0	y[58] = 0xa3014314	y[59] = 0x4e0811a1
y[60] = 0xf7537e82	y[61] = 0xbd3af235	y[62] = 0x2ad7d2bb	y[63] = 0xeb86d391

子区块读取的顺序（每一列含 0～15）

z[0,⋯,15] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
z[16,⋯,31] = [1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12]
z[32,⋯,47] = [5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2]
z[48,⋯,63] = [0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9]，

定义向左循环位移位之数：

s[0,⋯,15] = [7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22]
s[16,⋯,31] = [5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20]
s[32,⋯,47] = [4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23]
s[48,⋯,63] = [6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21]，

- 运算过程

主要运算分为 4 回合，其中一回合如图 8-4 所示，其中 + 表示 $(\bmod 2^{32})$ 的加法，而 <<< s 表左旋 s 位。

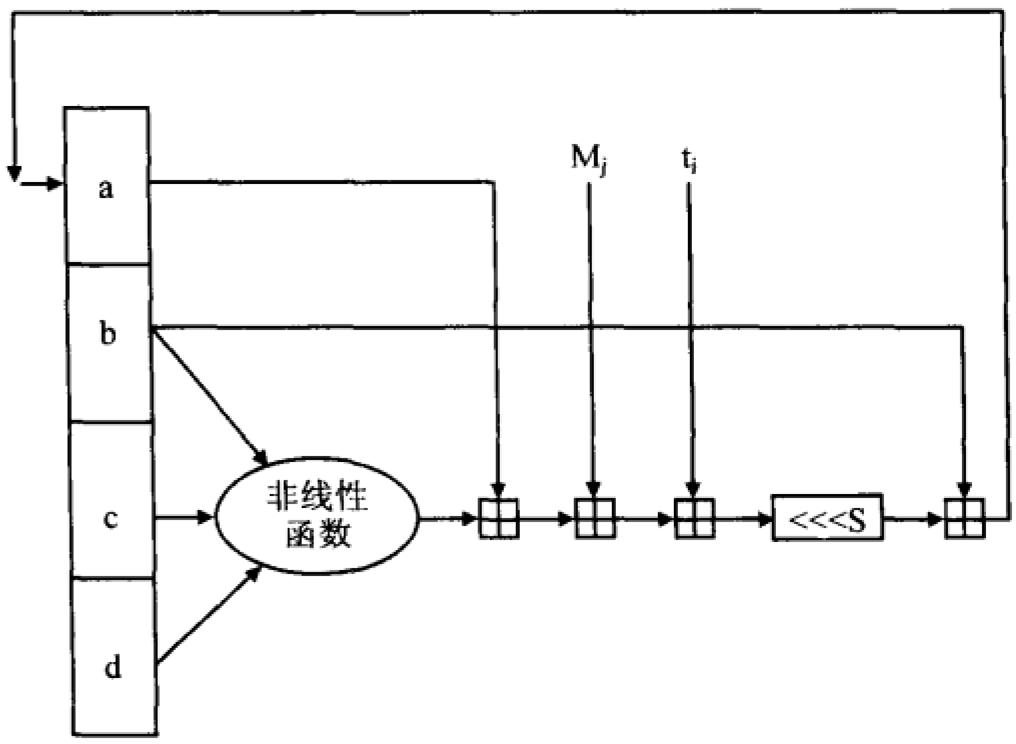


图 8-4　MD5 一回合作用

```
//将串联变数初始化
A=0x01234567;
B=0x89abcdef;
C=0xfedcba98;
D=0x76543210;
for(i=0;i<=L-1;i++)
{
    Y[i]=M[0]||M[1]||...||M[j]||...||M[14]||M[15];
    (a,b,c,d)=(A,B,C,D);
//回合 1:
    for(j=0;j<=15;j++)
    {
        t=(a+F(b, c, d)+M[z[j]]+y[j]);
        (a, b, c, d)=(d, b+(t<<<s[j]), b, c);
    };
//回合 2:
    for(j=16;j<=31;j++)
    {
        t=(a+G(b, c, d)+M[z[j]]+y[j]);
        (a, b, c, d)=(d, b+(t<<<s[j]), b, c);
    };
//回合 3:
    for(j=32;j<=47;j++)
    {
        t=(a+H(b, c, d)+M[z[j]]+y[j]);
        (a, b, c, d)=(d,b+(t<<<s[j]), b, c);
    };
//回合 4:
    for(j=48; j<=63;j++)
    {
```

```
        t=(a+I(b, c, d), M[z[j]], y[j]);
        (a, b, c, d)=(d, b+(t<<<s[j], b, c);
    };
//串联变数更新：
    (A, B, C, D)=(A+a, B+b, C+c, D+d);
}
```

- （输出）将最后所得 A||B||C 输出。

8.10 SHA-1 Hash 函数

美国国家标准局在制定 DSA 之后，在 1993 年也制定了使用 DSS 的 Hash 函数的标准，即安全 Hash 算法（Secure Hash Algorithm，SHA），其主要架构承袭了 MD 系列的 Hash 函数，所不同的是，MD 系列的 Hash 摘要大都是 128 位，而 SHA 的 Hash 摘要为 160 位，就计算安全性而言，当然是较长的 SHA 更好。稍后，于 1995 年推出了修订版，SHA-1，为 SHA-1 Hash 函数的算法。

算法 8.9（SHA-1 Hash 函数） 令 SHA-1 Hash 函数

$$h：\{任意长度的信息\} \to \{长度为 160 位的信息摘要\}$$

其中 m 为输入的信息，而输出的摘要

$$h(m)=A||B||C||D||E$$

A、B、C、D、E 均为 32 位的 $\mathbb{F}_2$ 字串，输出的摘要为 160 位，即 A、B、C、D、E 的字串联结。

- （常数以及函数的定义）SHA-1 在 4 回合中所用的非线性函数分别为 $f[i]\,(i=0,\cdots,79)$ 定义为

$$f[i](X,Y,Z)=\begin{cases}(X\wedge Y)\vee((\neg X)\wedge Z) & 若 1\leqslant i\leqslant 19\\ X\oplus Y\oplus Z & 若 20\leqslant i\leqslant 39\\ (X\wedge Y)\vee(X\wedge Z)\vee(Y\wedge Z) & 若 40\leqslant i\leqslant 59\\ X\oplus Y\oplus Z & 若 60\leqslant i\leqslant 79。\end{cases}$$

在算法中所采用的 4 个常数为

$$K_i=\begin{cases}0x5a827999, & 1\leqslant i\leqslant 19\\ 0x6edgeba1, & 20\leqslant i\leqslant 39\\ 0x8f1bbcdc, & 40\leqslant i\leqslant 59\\ 0xca62c1d6, & 60\leqslant i\leqslant 79。\end{cases}$$

其实这些常数分别来自于

$$2^{30}\times\sqrt{2}，\ 2^{30}\times\sqrt{3}，\ 2^{30}\times\sqrt{5}，\ 2^{30}\times\sqrt{10}$$

的整数部分。另外 5 个串联变量的初始值（MD5 用 4 个）分别为：

A = 0x67452301

B = 0xefcdab89

C = 0x98badcfe

D = 0x10325476

$E = 0xc3d2e1f0$ 。

- （数据初始添加）与 MD5 一样，先将原始数据 l 位的 m 添加成 512 倍数长的数据 m'，即

$$m' = m||1000\cdots0||m \text{ 原始长度} \pmod{2^{64}} 。$$

此时

$$m' = M[0]||M[1]||\cdots||M[L-1] ,$$

其中每一区块 M[i] 均为 512 位，而每一区块 M[i] 可再细分为 16 块 32 位的子区块，即

$$M[i] = W[0]||W[1]||\cdots||W[15] 。$$

- （运算过程）其中 + 表示 $\pmod{2^{32}}$ 的加法，而 <<<s 表左旋 s 位，主要运算分为 4 回合 80 次，其中 SHA-1 一回合作用如图 8-5 所示。

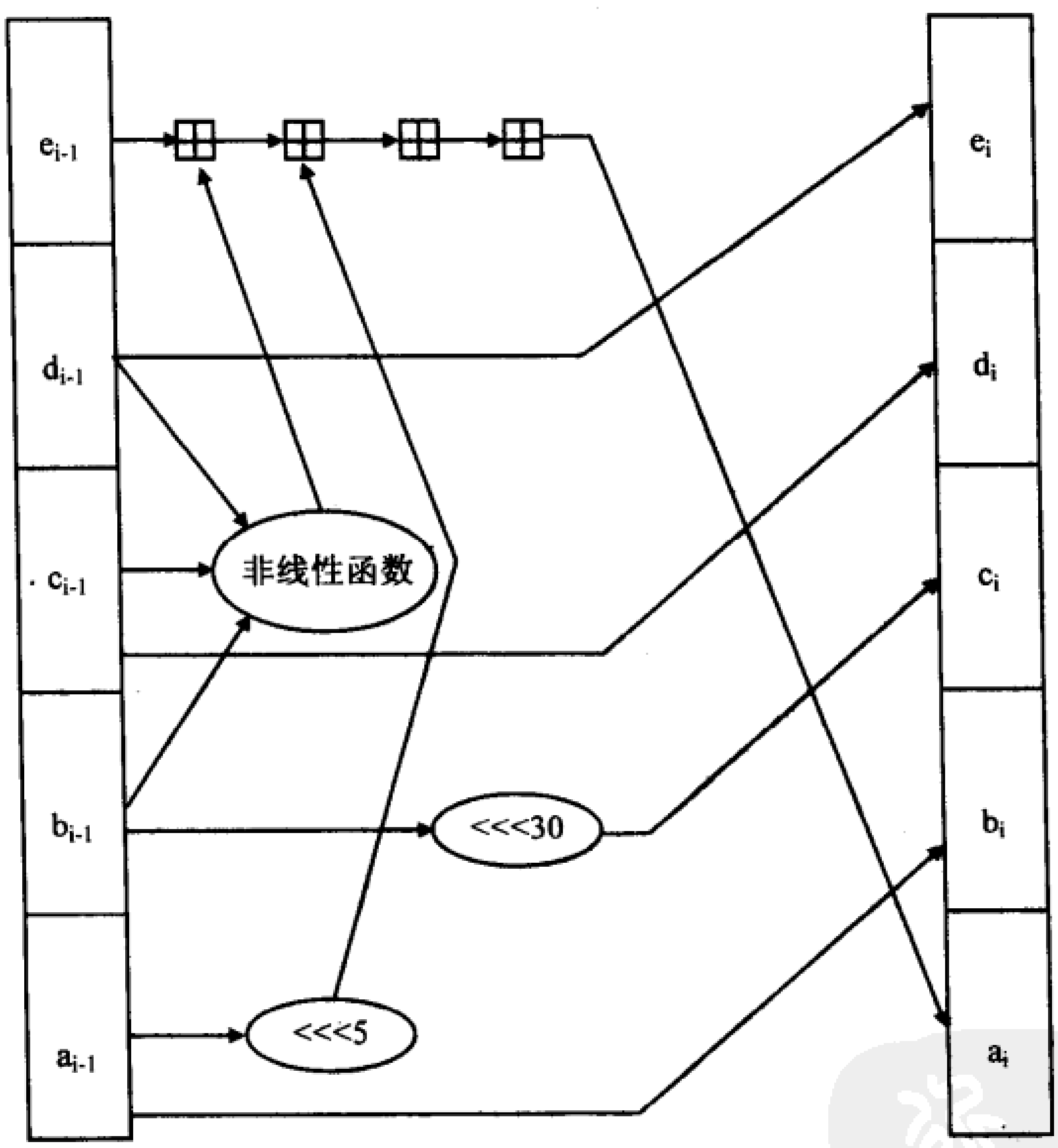


图 8-5　SHA-1 一回合作用

```
//先将暂存器初始化
A=0x67452301;
B=0xefcdab89;
C=0x98badcfe;
D=0x10325476;
E=0xc3d2e1f0;
for (i=0;i<=L-1;i++)
{
```

```
M[i]=W[0]||W[1]||....||W[15];

//定义子区块 W[16],..,W[79]:
for (t=16;t<=79;t++)
    W[t]=(W[t-3] XOR W[t-8] XOR W[t-16])<<<1);
a=A;b=B;c=C;d=D;e=E;
//主要运算分四回合 80 次:
for (t=0;t<=79;t++)
{
    TEMP=(a<<<5)+f[t](b, c, d)+e+W[t]+K_t;
    e=d;
    d=c;
    c=b<<<30;
    b=a;
    a=TEMP;
};

//串联变数更新:
(A, B, C, D, E)=(a+A, b+B, c+C, d+D, e+E);
};
```

- （输出信息）最后输出

$$h(m) = A \| B \| C \| D \| E \text{。}$$

比较 MD5 与 SHA-1 两者的区别，可以发现：

（1）SHA-1 的信息摘要为 160 位要比 MD5 的 128 位安全。借助于生日攻击，有效成功攻击的计算复杂度分别为

$$O(2^{80}) \text{与} O(2^{64}) \text{。}$$

（2）SHA-1 的基本运算为 80 次，而 MD5 四回合运算为 64 次，因此 SHA-1 的运算速度较 MD5 慢。

（3）两者都采用非线性函数，均为 4 个。

（4）SHA-1 演算中采用了 4 个常数，而 MD5 采用了 64 个，即

$$[\, 2^{32} \,|\sin(i)|\,] \quad (i = 1, \cdots, 64) \text{。}$$

（5）两者都很简易，不需要大量内存，也不需要大量计算时间，都可直接将其算法程序化。

另外值得一提的是，在本世纪初，在 2001 年美国国家标准局公告了另外三种新的 Hash 函数，即 SHA-256、SHA-384 以及 SHA-512，其中的代号就表示其摘要的位数大小，其算法与 MD 系列和 SHA-1 类似，也采用多次迭代方式运算，所用的常数也是来自简易算术的结果，非线性函数也是采用布尔代数运算而成。Hash 函数的发展，伴随着数字签名算法密钥长度的增加而增加其输出摘要的长度。

8.11 信息校验码 MAC

使用密钥的 Hash 函数，又称为信息校验码（Message Authentication Code，MAC），常用的方法就是使用任意区块密码算法，以 CBC 模式加密，只保留最后一个区块所产生的密文当

作 Hash 值，MAC 如图 8-6 所示。

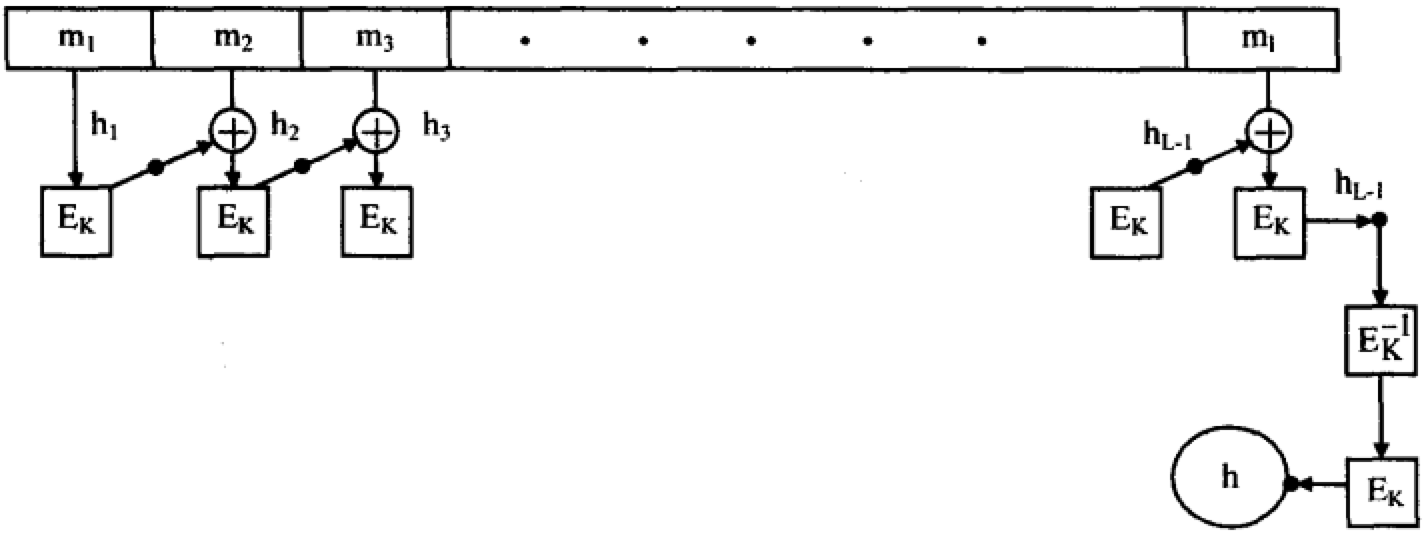


图 8-6　CBC-MAC

算法 8.10（CBC-MAC）

输入≫信息 m，特定区块密码算法 E 以及密钥 K。

输出≪n 位 m 的 MACh（n 为 E 的区块位长度）。

（1）将信息 m 分割成以 n 位大小为单位的区块，并将最后一区块填补数据

$$m = m_1 \| m_2 \| \cdots m_L\text{。}$$

（2）（CBC 模式加密）令 E_K 为密钥 K 的加密函数，计算

$$h_1 = E_K(m_1)$$

$$h_j = E_K(h_{j-1} \oplus m_j)\ ,\ \ (j = 2,3,\cdots,L)$$

注意，此时初始向量 $IV = 0$，只需要保留 h_L 值，其余的 $h_j\ (j = 1,\cdots,L-1)$ 皆无须保留。

（3）（增强 MAC 的选择性过程）另外选取别的密钥 $K' \neq K$，计算

$$h = E_K(E_{K'}^{-1}(h_L))\text{。}$$

MAC 本身是加上密钥的 Hash 函数，所以 Hash 函数的性质都会成立，这在提供信息完整性上有实用价值。另外 MAC 也不一定要借助区块加密算法来实现，其实也可以用常见的 Hash 函数，如 MD5、SHA-1 或是 RIPEMD 完成，这就是以 Hash 为基础的 MAC，如以下例子所示。

例 8.9（HMAC）　令 h 为一抗碰撞的 Hash 函数，K 为此 MAC 的密钥，m 为信息，则可造成一个 MAC 如下：

$$HMAC = h(K \| p_1 \| h(K \| p_2 \| m))$$

其中 p_1、p_2 为不同字串用来填补成完整的区块。

第 9 章　质数与大整数算术

在公开密钥密码系统的计算中，动辄百位整数的算术计算，加、减、乘、除以及模运算是当中最基本的。另外公开密钥密码系统需要大质数，如强质数、安全质数等，这些问题都牵涉到公开密钥密码系统的实际操作问题。对于习惯 Java 程序设计的读者，由于有 BigInteger Class，在撰写公开密钥密码程序时，可以省去不少开发底层计算工具的工作，在此我们也特别对此作一简介；另外网络资源也有不少有关计算数论的工具，部分基本计算如加、减、乘、除也有以组合语言开发的工具，这些也将在本章中加以介绍。

9.1　大整数的加减乘法

在本节中，我们将介绍大整数的加、减、乘法，这些运算大致与小学所学的“直式”算法一致，不论是用何种程序设计技巧，其算法都大致相同，目前主流电脑大都是 32 位机器，以 C 语言为例，unsigned long 型数据可处理 0 至 $2^{32}-1$ 的非负数，十进位的数字表达在电脑上的计算并不自然，反而是 2 进位、16 进位较为自然，无论加、减、乘法，都会涉及进位、借位问题，在此采用 unsigned short 为单位，即以 2^{16} 为基底，因此每一个非负大整数，都可以用 unsigned short 序列的方式处理，如

$$x = x[L-1]2^{16(L-1)} + x[L-2]2^{16(L-2)} + \cdots + x[2]2^{32} + x[1]2^{16} + x[0]$$

$$y = y[L-1]2^{16(L-1)} + y[L-2]2^{16(L-2)} + \cdots + y[2]2^{32} + y[1]2^{16} + y[0],$$

先考虑 $z = x + y$，其中 z 可表为

$$z = z[L]2^{16L} + z[L-1]2^{16(L-1)} + \cdots + z[2]2^{32} + z[1]2^{16} + z[0],$$

其 C 语言核心程序如下：

程序 9.1　计算 $x + y = z$：

```
unsigned int carrier=0;
for(unsigned short i=0;i<L;i++)
{
    unsigned int w=(unsigned int)x[i]+
         (unsigned int)y[i]+carrier;
    if(w>0xffff)
         {
             carrier=1;
             z[i]=(unsigned short)(w-0x10000);
         }
    else
         {
             carrier=0;
             z[i]=(unsigned short)w;
         };
```

```
};
z[L]=(unsigned short)carrier;
```

假设 $x>y$，则 $d=x-y$ 可表示为

$$d=d[L-1]2^{16(L-1)}+d[L-2]2^{16(L-2)}+\cdots+d[2]2^{32}+d[1]2^{16}+d[0],$$

如同小学直式算式，其 C 程序核心为：

程序 9.2　计算 $x-y=d$：

```
unsigned int carrier=0;
for(unsigned short i=0;i<L;i++)
    {
        int w=(unsigned int)x[i]-(unsigned int)y[i]
            -carrier;
        if(w<0)
            {
                carrier=1;
                d[i]=(unsigned short)(w+0x10000);
            }
        else
            {
                carrier=0;
                d[i]=(unsigned short)w;
            };
};
```

以上的计算过程均是以 $2^{16}=65536$ 为基底，若读者仍执意要用十进位表达，又不想花精神撰写不同进位之间转换的程序，可以考虑以 10^9 为底，（因为 $2\times10^9<2^{32}$）略微修改即可。对于乘法 $m=x\times y$，其中 m 可表示成

$$m=m[2L-1]2^{16(2L-1)}+m[2L-2]2^{16(2L-2)}+\cdots+m[2]2^{32}+m[1]2^{16}+m[0],$$

C 程序核心如下：

程序 9.3　计算 $x\times y=m$：

```
unsigned int carrier=0;
for (unsigned short i=0;i<2*L-1;i++)
    {
        unsigned int w=carrier;
        for(unsigned short j=0;j<=i;j++)
            w=w+(unsigned int)x[j]*(unsigned int)y[i-j];
        if(w>0xffff)
            {
                carrier=(unsigned short)w/0x10000;
                m[i]=(unsigned short)(w-carrier<<16);
            }
            //REM: carrier<<16=carrier*0x10000
        else
            {
                carrier=0;
```

```
                m[i]=(unsigned short)w;
            };
        };
    m[2*L-1]=(unsigned short)carrier;
```

加、减法都是在线性时间内可完成，O(L) 次计算即可，而乘法则需要 $O(L^2)$ 次加、减运算。另外较为有效的乘法算法，在实际操作中，便很值得考虑以提高效率，一种称为 Karatsuba 乘法的做法如下：令整数 x 及 y 均为 2L 位，即 2^{2L}，则 x 与 y 可表示为

$$x = x_1 2^L + x_0$$
$$y = y_1 2^L + y_0$$

其中 $0 \leqslant x_0, x_1, y_0, y_1 \leqslant 2^L$。先计算

$$A = x_0 y_0$$
$$B = (x_0 + x_1)(y_0 + y_1)$$
$$C = x_1 y_1$$

则

$$\begin{aligned} x \times y &= (x_1 2^L + x_0)(y_1 2^L + y_0) \\ &= x_1 y_1 2^{2L} + (x_1 y_0 + x_0 y_1) 2^L + x_0 y_0 \\ &= C2^{2L} + (B - A - C)2^L + A \end{aligned}$$

这样一次 2L 位的乘法，便可化为 3 次 L 位的乘法与 3 次 2L 位的加减法及 2 次 L 位的加减法，以 M(L) 代表 L 位乘法计算次数，而 A(L) 代表 L 位加、减法的计算次数，则

$$M(2L) = 3M(L) + 2A(L) + 3A(2L),$$

因为 $A(L) \approx n$，故得 $M(2L) \approx 3M(L) + 8L$；若 L 位的乘法也如法炮制，即化为 3 个 $\lceil L/2 \rceil$ 位的乘法与若干加减法，$\lceil L/2 \rceil$ 位的乘法化为 3 个 $\lceil \lceil L/2 \rceil /2 \rceil$ 位的乘法与若干加减法，依此类推，可得

$$M(L) = O(n^{\log_2 3}) = O(n^{1.585\cdots}),$$

其计算复杂度可借此降低，算法可以用递归的方式呈现。

算法 9.1（Karatsuba 乘法）

输入≫ 2 个均为 L 位的大整数 x 及 y。

输出≪ x×y 的值。

```
KarastubaMultiply(x,y)
{
    L=max(x.bitlength,y.bitlength);
    if(L>1)
        {
            LL=ceil(L/2);
            x=x[1]*2^LL+x[0];//将 x,y 分割成 2 半
            y=y[1]*2^LL+y[0];
            A=KarastubaMultiply(x[0],y[0])
            B=KarastubaMultiply(x[0]+x[1],y[0]+y[1]);
            C=KarastubaMultiply(x[1],y[1]);
```

```
            return(C,B-A-C,A);//C*4^LL+(B-A-C)*2^LL+A
        }
    else
            return x*y; //L=1
}
```

9.2　大整数的除法

四则运算中，除法是最麻烦的，在多位数的除法中，商数各数字的估计，必须先用乘法试算，往往所“猜测”的商数过大，以十进位为例。

例 9.1　正整数 $a = 1345$ 除以 $b = 297$，首先比较两数的首位数 $a_3 = 1 < 2 = b_2$，所以商数为 1 位数，然后以被除数 a 的前两位数值 $a_3 \times 10 + a_2 = 13$ 试除除数 b 的首位数 $b_2 = 2$ 得

$$[13/2] = 6,$$

但

$$297 \times 6 = 1782 > 1345,$$

所猜测的商过大，而且

$$297 \times 5 = 297 \times (6-1) = 1782 - 297 = 1485 > 1345,$$

商值必须再向下修正，以乘法式算得

$$297 \times 4 = 297 \times (5-1) = 1485 - 297 = 1188 < 1345,$$

而得商值 $q = 4$，而余数为

$$1345 - 1188 = 157。$$

例 9.2　正整数 $a = 485$ 除以 $b = 29$，首先比较两数的首位数 $a_2 = 4 > 2 = b_1$，所以商数为 2 位数 $q = 10 \times q_1 + q_0$。

（1）（求 q_1）以 a 的首位 $a_2 = 4$ 试除以 b 的首位 $b_1 = 2$ 得

$$[4/2] = 2,$$

猜测 $\hat{q}_1 = 2$，但

$$29 \times 2 = 58 > 38,$$

过大，而

$$29 \times 1 < 38,$$

故 $q_1 = 1$。

（2）（求 q_0）先计算

$$a - (b \times q_1) \times 10 = 385 - (29 \times 1) \times 10 = 75 = a',$$

以 a' 取代 a 的位置，用 a' 的首位数 $a_1' = 7$ 试除 b 的首位 $b_1 = 2$ 得

$$[7/2] = 3,$$

猜测 $\hat{q}_0 = 3$，但

$$29 \times 3 = 87 > 75$$

过大，而

$$29 \times 2 = 29 \times (3-1) = 87 - 29 = 58 < 75,$$

故 $q_0 = 2, q = 12$。

（3）（求余数 r）计算

$$r = a' - q_0 \times b = 75 - 2 \times 29 = 17 \text{。}$$

由上述两例中，与“直式”除法类似，主要是考虑 n+1 位或 n 位数的被除数

$$a = a_n a_{n-1} \cdots a_1 a_0 \quad (a_n \neq 0 \text{或} a_n = 0)$$

除以 n 位数的除数

$$b = b_{n-1} b_{n-2} \cdots b_2 b_1 \quad (a \leq b),$$

以求出商数的各项数值，而所猜测商值为

$$\hat{q} = \min([\frac{10 \times a_n + a_{n-1}}{b_{n-1}}], 9),$$

其中 $\hat{q}$ 与真实 q 值满足

$$\hat{q} - 2 \leqslant q \leqslant \hat{q},$$

猜测值不会比真实值 q 小，但最多大 2；注[1]而若商值有多位数时，做法与直式除法一致，只需要逐次以 a′ 取代 a，最终会得到商数以及余数。在程序设计时，考虑以 2^{16} 为单位，取代十进制，假设正整数

$$a = a[n+m-1]2^{16(n+m-1)} + \cdots + a[1]2^{16} + a[0]$$

$$b = b[n-1]2^{16(n-1)} + \cdots + b[1]2^{16} + b[0]$$

以 a 除以 b 得商数 q 与余数 r，其中

$$q = q[m]2^{16m} + \cdots + q[1]2^{16} + q[0]$$

$$r = r[n-1]2^{16(r-1)} + \cdots + r[1]2^{16} + r[0] \text{。}$$

可将除法的算法整理如下。

算法 9.2（除法） 除法计算商与余数：

```
//(n+1)-byte 整数 x 除以 n-byte 整数 y 的算法:
division(x,y)
{
    x=(x[n],x[n-1],...,x[1],x[0]);
    y(y[n-1],y[n-2],...,y[1],y[0]);
    q_test=min(x[n]*(2^16)+x[n-1] / y[n-1],2^16-1);
    temp=q_test*y;
    while(temp>=x)
        {
            q_test--;
            temp=temp-y;
        };
    q=q_test;
    remainder=x-q*y;
    return(q,remainder)
};
```

注[1] 可参考 Knuth 的经典名著“The Art of Computer Programming” Vol 2,3rd Ed.Addison-Wesley，(1998) 的第 4.3.1 节。

```
//主要算法部分:
a=(a[n+m-1},...,a[1],a[0]);
b=(b[n-1],...,b[1],b[0]);
if(a[n+m-1]>=b[n-1])
    {
        a=(0,a[n+m-1],...,a[1],a[0]);
        m++
    };
x=(a[n+m-1],a[n+m-2],...,a[m-1]);
for(i=m;m>=2;m--)
    {
        (q[i],remainder)=division(x,b);
        x=remainder*(2^16)+a[i-2];
    };
(q[0],r)=division(x,b);
```

除法是在四则运算中时间花费最多的，因此在密码系统程序设计中要尽量避免。

9.3　Montgomery 算术

在公开密钥密码的计算中，要大量用到模运算，特别是模除法，其实就是要以广义辗转相除法计算乘法反元素，再进行乘法，直观地看是要先进行多次乘法，然而 Peter L. Montgomery 在 1985 年的一篇文章中注[2]指出其实是可以避开惹人烦恼的除法运算的，他主要的观察是基于下列非常简单的模运算事实。

性质 9.1　有两个互质的整数 n 和 r，令 r^{-1} 表示 r 在 (mod n) 模运算中的乘法反元素，n^{-1} 表示 n 在 (mod r) 模运算中的乘法反元素。定义

$$n' = -n^{-1} \pmod r$$
$$m = tn' \pmod r \ ,$$

其中 t 为某整数，则

$$\frac{t+mn}{r} \equiv tr^{-1} \pmod n \text{。}$$

证明：$\dfrac{t+mn}{r} \equiv \dfrac{t+ntn' \pmod r}{r}$ （其中 r′ 为某整数，且 $rr' \equiv 1 \pmod n$ ） $\equiv tr^{-1} \pmod n$ 。

另外，虽然同是在 $\mathbf{Z}/n$ 上考虑问题，Montgomery 考虑与

$$\{0,1,2,3,\cdots,n-1\}$$

不同的代表系统，即

$$R = R(r,n) = \{ir \pmod n \mid 0 \leqslant i \leqslant n\} \ ,$$

其中 $\gcd(r,n)=1$；在实际中一般选取

$$r = 2^s > 0 \text{ 使得 } 2^{s-1} \leqslant n \leqslant 2^s \text{。}$$

注[2] Montgomery,Peter L, “Modular mulliplication without trial divesion”, Mathematics of Computation, P.519-521,44(170),(1985)。

因此在代表系统中乘法的做法为：

定义 9.1 令 $\gcd(r,n)=1$，令 $R(r,n)=\{ir \pmod n \mid 0\leqslant i\leqslant n\}$ 是 $\mathbb{Z}/n$ 上的代表系统。当 a，$b\in R(r,n)$ 时，其"乘法"可定义为

$$a\boxtimes b := abr^{-1} \pmod n。$$

诚然，$a=ir$，$b=jr$，所以

$$a\boxtimes b=(ir)(jr)r^{-1}\equiv(ij)r\in R(r,n),$$

定义上是没有问题的。值得注意的是 r^{-1} 的计算可直接由广义辗转相除法或 xeudidean() 求得，计算成本较高；然而当 n 为奇数时，取 $r=2^s$ 满足 $2^{s-1}\leqslant n<2^s$，计算更为简捷，如以下算法。

算法 9.3（Montgomery 乘法）

输入≫n 为奇正整数，$r=2^s$，使得 $2^{s-1}\leqslant n<2^s$，a，$b\in R(r,n)$。

输出≪Montgomery 乘积 $a\boxtimes b=abr^{-1} \pmod n$。

```
InverseMod2power(n,s)
//计算 n'=n^(-1)(mod 2^s)
//其中 n 为奇数
{
    result=1;
    for(i=2;i<=s;i++)
        if(2^(i-1)<(x*result)%2^i)
            result+=2^(i-1);
        return result;
}
MontgomeryProduct(a,b,n)
{
    t=ab;
    //r=2^s:
    s=ceil(log(n)/log(2))+1;

    //用 InverseMod2power(n,s)计算 n'=n^(-1)(mod 2^s):
    n'=InverseMod2power(n,s);
    m=t*n';
    u=(t+m*n)>>s;
    //(t+m*n)/r

    if(u>=n)
        u=-n;
    return u;
}
```

至于要计算 $product \equiv xy \pmod n$ 也可利用 Montgomery 乘法，即

```
x=x*r%n;
product=Montgomeryproduct(x,y,n);
```

当然这在单独计算 $xy \pmod n$ 时并无多大意义。然而对计算大批同模数 n 的模乘法运算时，如模指数运算，就可借此加快运算速度。

9.4　Miller-Rabin 质数测试

本书所讨论的公开密钥密码技术都需要质数，质数分布理论表明，[注3]随机选取一个 1024 位正奇数且为质数的概率约为

$$\frac{2}{\ln(2^{1024})} \approx \frac{1}{355},$$

（大于 2 的偶数当然不是质数）因此我们需要判断该数是否是质数，一种简易的方法就是利用费马小定理：

"令 a 为任意与 n 互质的整数，若 n 为质数 $\Rightarrow a^{n-1} \equiv 1 \pmod n$"

当然只要能找出某值 a 使得该式不成立，n 就一定不是质数。但是，即使该式对所有与 n 互质的 a 都成立，也不能保证 n 就一定是质数。因此就算 n 能够通过费马小定理的测试，也只可能为合数。

例 9.3　$n = 561 = 3\times 11\times 17$，由中国余式定理

$$f:\mathbb{Z}/561 \cong \mathbb{Z}/3\times\mathbb{Z}/11\times\mathbb{Z}/17$$

$$x \mapsto (x_1 = x \bmod 3, x_2 = x \bmod 11, x_3 = x \bmod 17)$$

所以

$$x^{561} = f^{-1}(x^{561} \bmod 3, x^{561} \bmod 11, x^{561} \bmod 17)$$
$$= f^{-1}(1,1,1) = 1 \pmod{561}$$

但稍微将费马小定理推广，可得下例结果：

定理 9.1　令 n 为奇数，且 $n-1 = d2^s$，其中 d 为奇数，若 a 为与 n 互质的整数，若 n 为质数，则以下两个陈述中必有一个成立：

- $a^d \equiv 1 \pmod n$；
- 或存在某整数 r（其中 $0 \leqslant r \leqslant s-1$）使得

$$a^{2^r d} \equiv -1 \pmod n。$$

证明：由于 n 为质数，$\mathbb{Z}/n[x]$ 为惟一分解环 UFD，可考虑多项式 $f(x) = x^{n-1} - 1$ 的因式分解：

$$\begin{aligned} x^{n-1} - 1 &= (x^d)^{2^s} - 1 \\ &= ((x^d)^{2^{s-1}} + 1)((x^d)^{2^{s-1}} - 1) \\ &= ((x^d)^{2^{s-1}} + 1)((x^d)^{2^{s-2}} + 1)((x^d)^{2^{s-2}} - 1) \\ &\ \vdots \\ &= ((x^d)^{2^{s-1}} + 1)((x^d)^{2^{s-2}} + 1)((x^d)^{2^{s-3}} + 1)\cdots(x^d + 1)(x^d - 1) \end{aligned}$$

考虑 $f(a) = a^{n-1} - 1$，因费马小定理 $f(a) \equiv 0 \pmod n$，故上式的质因式中必有一个 $\equiv 0 \pmod n$，即定理的结论成立。

注 9.1　与费马小定理一样，若整数 n 与所有满足 $\gcd(a,n) = 1$ 的 a 都满足上述定理的结论，

[注3] 参见本书的"质数理论"一节。

并不表示 n 就一定是质数。但经进一步分析可知，定理的结论成立但不为质数的概率最多为 $\frac{1}{4}$。

现行公开密钥密码系统最常用的质数测试方式，即所谓 Miller-Rabin 质数测试，以类似 C 程序代码的方式呈现：

算法 9.4（Miller-Rabin 质数测试） 输入大整数 n，判断是否为质数。

```
bool MillerRabin_Test(biginteger n,biginteger a)
{
// a=Random_Integer(2,n-1);
    if(gcd(a,n)>1)
        return 0;//0 表示一定不是质数
    biginteger s=1;
    biginteger d=(n-1)/2;
    while(is_even(d))
        {
            d/=2;
            s++;
        }; //n=2^s*d+1
    if(gcd(a,n)!=1)
        return 0;//×_Î²ó
    else
        if(((a^d)-1)%==0)
            return 1;
        else
            {
                for(biginteger r=0;r<s;r++)
                    if((a^(2^r*d)+1)%n==0)
                        return 1;
                    //1 表示可能为质数概率为 1-1/4
                return 0;//一定不是质数
            };
}
```

如何使用 Miller-Rabin 质数测试找出适当的质数，这可在下面的算法中执行：

算法 9.5 输出一个 k 位质数 x。

```
Miller_Rabin_Probable_Prime()
{
Step1: x=Random_Integer(1,b[k-2],b[k-3],...,b[2],b[1],1);
        //二进位数，b[i]=0 或 1，x 为一奇数
Step2: for(i=1; prime[i]<=B; i++)//prime[i]表第 i 个质数
        if ( x%prime[i]==0)
            goto Step1;//x 非质数，重来！
Step3: for(i=1;i<=t;i++)
        {
            a[i]=Random_Integer(2,x-1);
            if ( MillerRabin(x,a[i])==0)
                goto Step1;
        };
    output<<x<<" 可能是质数,";
```

```
    output<<" 所有<= "<<B<<" 的质数无法整除,";
    output<<" 并通过 Miller-Rabin 测试 "<<t<<" 次数!";
}
```

注 9.2

（1）在 Step2 中，可由试除筛去非质数，通过此测试的奇整数 x 的比率为

$$1-\prod_{\substack{p:\ \text{质数} \\ 2<p\leqslant B}} 1-\frac{1}{p}。$$

（2）若省略 Step2，则所输出的数为质数的概率为

$$\text{prob}(x=\text{质数})\geqslant 1-\left(\frac{1}{4}\right)^{t}。$$

（3）若取 $B=10^6$， $t=3$ ，则

$$\text{prob}(x=\text{质数})\geqslant 1-\left(\frac{1}{2}\right)^{80}，$$

其中 x 为 k 位的奇数， $k\geqslant 1000$ 。

（4）若假设大多数学家都相信广义 Riemann 猜想（Generalized Riemann Hypothesis），则此算法在多项式时间次测试后，所输出的 x 为质数，此时 $t=O((\log(n))^2$ 。

9.5 Agrawal-Kayal-Saxena 算法

纽约时报（New York Times）在 2002 年 8 月 8 日报导了“新的方法解开了数学的关键问题”，即印度科技学院的 Manindra Agrawal 、Neeraj Kayal、Nitin Saxena 师徒三人提出一个能在多项式时间内，判断任意正整数是否是质数的算法，注[4]该算法相当干净且漂亮，这个结果轰动了数学界与信息界。对于数学界而言，自从高斯以来，判断一个整数是否为质数，本来就是许多数学家所关心的问题；对于信息界而言，由于大多数的公开密钥密码技术需要质数，而且是大质数，一个能真正在适当时间内，找到真正质数的 Deterministic 算法，是关键性的问题，这样的算法的实际价值是不言而喻的。曾经是 1997 年国际数学奥林匹克竞赛印度代表队队员的 Kayal 与 Saxena 选择了信息系就读，并参加了 Agrawal 带领下的学生专题计划，在 2002 年夏天开始博士生研究。在 2002 年 6 月 10 日，他们的研究有了突破性进展，他们找出了对质数幂次的重要刻画，即

定理 9.2（Agrawal-Kayal-Saxena） 令 $n\in\mathbb{N}$ 为自然数且 $s\leqslant n$ 。令质数 q 与 r 满足

$$q \mid (r-1)$$

$$n^{\frac{r-1}{q}} \not\equiv 0,1 \pmod r$$

$$C_s^{q+s-1} \geqslant n^{2[\sqrt{r}]}。$$

若对所有的 $1\leqslant a\leqslant s$ 都满足

注[4] Agrawal,M and Kayal,N and Saxena,Nitin,“PRIMES is in P”,Preprint of Aug.2002，http://www.cse.iitk.ac.in/news/primality.html。

（1）a 与 n 互质。

（2）且在多项式环 $\mathbf{Z}[x]$ 中 $(x-a)^n \equiv x^n - a \mod(x^r-1,n)$。

则 n 为某质数的幂次。

在此我们略去证明。值得一提的是，证明主要用到一般基础数论的结果，如费马小定理等，对于具有代数以及基础数论知识的读者而言，要理解该定理的证明并非难事。注[5]将该结果整理一下，就可得到下列判断质数的算法.

算法 9.6（Agrawal-Kayal-Saxena 算法） 输入一个正整数 $n>1$，并判断其是否为质数。

```
bool Agrawal_Kayal_Saxena(biginteger n) //n>1 整数
{
Step 1: if(is Power(n)) //若 n=a^b,其中 b>1
            return 0; // n 为某数幂次，非质数
Step 2: r=min{r | ord(n,Z/r)>4*(log(n)^2)};
    // ord(n,Z/r)表示 n 在乘法群 Z/r 的秩
    for(a=2;a<=r;a++)
        {
            gcd=gcd(a,n);
            if(gcd>1&&gcd<n)
                return 0; //n 为合数，非质数
        };
    s=floor(2*sqrt(Eulerphi(r))*log(n));
step3: for(a=1;a<=s;a++)
        if((x+a)^n!=x^n+a(mod x^r-1,n))
            return 0; // n 为合数，非质数
    return 1; //n 为质数
};
```

该算法主要可分成三个步骤，其中：

- 要判断 n 是否为某数的幂次，可在多项式时间内完成检查。
- 而在 Step 2 中 r 与 s 的选取，满足于定理中的假设的，这在解析数论中是常考虑的问题。
- 主要步骤 Step 3 是本算法最花时间的，在不假设任何数论上重要猜想的情况下，其计算复杂度为 $O(\log^{7.5+\varepsilon} n)$，其中 ε 为任意小的正数，故本算法的计算复杂度保守估计为 $O(\log^{7.5+\varepsilon} n)$。

然而若假设 Hardy 与 Littlewood 关于 Germain 质数分布的猜想成立：

$$\#\{q \leqslant x \mid q\text{与}2q+1\text{ 为质数}\} \approx \frac{2Cx}{\ln^2 x},$$

其中 C 为某常数：此时算法的计算复杂度可降至 $O(\log^{6+\varepsilon} n)$，但若更进一步假设。

猜想 9.1 令 r 为质数且 $r \mid n$。若

$$(x-1)^n = x^n - 1 \mod (x^r - 1, n),$$

注[5] 可参阅 Bornemann，F 的“ PRIMES Is in P:A Breakthrough for ‘Everyman’。”Notices of the AMS, May 2003 一文，或直接参阅 Agrawal、Kayal 以及 Saxena 的 Preprint。

则 n 为质数或 $n^2 \equiv 1 \pmod r$。

则算法的计算复杂度可降低至 $O(\log^{3+\varepsilon} n)$。若真如此，Agrawal-Kayal-Saxena 算法的实用价值就非常大。由于这项重大的成果，有人预期 Agrawal 应获数学界每四年一度的最大殊荣 Fields 奖，然而在 2002 年 10 月 30 日，Agrawal 就已获 Clay 研究奖，这项奖项也曾经颁给证明费马最后猜想的 Andrew Wiles 以及上届 Fields 奖获奖者 Connes、Lafforgue、Witten 等人。

9.6　公开密钥密码的质数

在不同的公开密钥密码系统中，因安全性的需求，往往需要不同结构的质数；如 RSA 密码系统，不少人就建议用强质数，ElGamal 密码以及 ElGamal 数字签名用安全质数，而 DSA 质数的产生也由 NIST 制定。

9.6.1　强质数

所谓强质数就是不易被 Pollard 的 $p-1$ 法或是 William 的 p+1 法攻击成功的 RSA 质数，即：

定义 9.2（强质数，Strong Prime）　当正整数 r、s、t 存在满足下列条件时，质数 p 称之为强质数：

- $p-1$ 拥有大质数因子 r。
- $p+1$ 拥有大质数因子 s，
- $r-1$ 拥有大质数因子 t。

甚至有人建议 $\frac{p-1}{2}$ 应为质数，如此 r 与 s 之存在，自动成立。

强质数可用 Gordan 算法迅速找出。注[6]

算法 9.7（Gordan 算法）　输出≪强质数 p。

```
Gordan()
{
    s=RandomPrime;
    t=RandomPrime;// s 与 t 的位数大约相同
    i=RandomInteger;
    while (isPrime(r=2*i*t+1)==0)
        i++;
    p0=2*(s^(r-2)%r)*s-1;
    j=RandomInteger;
    while (isPrime(p=p0+2*j*r*s)==0)
        j++;
    return p;
}
```

注意：一般随机质数的找法还是用 Miller-Rabin 算法。

证明：如此找到的 p 确实是强质数，其中 $p = p_0 + 2jrs$ 是质数。利用费马小定理，得

注[6] ohn A.Gordan 的"Strong primes are easy to find"in Proceedings of EUROCRYPT 84,P.216-223, Paris,(1985)Springer-Verlag,LNCS.209。

$$p_0 = 2(s^{r-2} \pmod r)s - 1 \equiv 2\times 1 - 1 = 1 \pmod r,$$

$$p_0 = 2(s^{r-2} \pmod r)s - 1 \equiv -1 \pmod s。$$

检查强质数条件：

- $p-1 = p_0 + 2jrs - 1 \equiv 0 \pmod r$，$r = 2it+1$ 是 $p-1$ 的大质数因子。
- $p+1 = p_0 + 2jrs + 1 \equiv 0 \pmod s$，s 是 $p+1$ 的大质数因子。
- $r-1 = 2it \equiv 0 \pmod t$，t 是 $r-1$ 的大质数因子。

ElGamal 密码以及 ElGamal 使用安全质数，请参阅本书的"Pohlig-Hellman 密码与离散对数"一节。

9.6.2 DSA 质数

DSA 的质数 p 与 q 满足：

- $2^{159} < q < 2^{160}$，q 为 160 位的质数。
- $2^{L-1} < p < 2^{L}$，其中 $L = 512 + 64d$，而 $d = 0,1,2,\cdots,8$。
- $q \mid p-1$。
- 同时希望 $\frac{p-1}{2q}$ 应有大质数因子。

而 DSA 的质数也由 NIST 所提的算法产生：

算法 9.8（DSA 质数）：

输入≫整数 d，（$0 \leqslant d \leqslant 8$）。

输出≪一个 160 位的质数 q，以及 L 位的质数 p，其中

$$L = 512 + 64l,$$

$$q \mid p-1。$$

```
DSA_Prime(d)
{
    L=512+64*d;
    n=(L-1)/160;
    b=(L-1)%160;
    while(1)
    {
        do
    {
        s=RandomInteger(g-bit);
        //g>=160
        h=SHA-1;
        u=h(s) XOR h((s+1)%(2^g));
        q=1<<(g+1)+u<<1+1;
    } while(!isProbablePrime(q));
    // 使用 Miller-Rabin 测试至少 18 次
    for(i=0,j=2; i<4096;i++,j+=n+1)
    {
        W=0;
```

```
        for(k=0,k<=n,k++)
        {
            V[k]=h((s+j+k)%(2^g));
            if (k==n)
                V[k]=V[k]%(1<<b);
            W=W+V[k]*(1<<(160*k));
        };
        X=W+(1<<(2*L));
        c=X%(2*q);
        p=X-(c-1);//p%(2*q)==1
        if (p>=(1<<L-1))
        // 使用 Miller-Rabin 测试至少 5 次
        if (isProbablePrime(p))
        return (p,q);
    };
}
```

9.7 Java 的 BigInteger Class

在 Java 程序语言中，提供了 BigInteger Class 套件，注[7]这对有关公开密钥密码系统程序设计有相当大的帮助，在此套件中提出对大整数计算的底层算法，如加、减、乘、除以及模运算，甚至在寻找质数上，将 Rabin-Miller 算法写成单一命令，这些封装的套件命令，虽然不一定使用最佳的算法，但解决了底层计算问题，使得想要在 Java 上开发密码系统的设计者，只需要关注密码算法，在使用上非常便利。与一般 Java Class 一样，有其构造方法（Constructor）、方法（Method）以及变量，常用的有下列各项。

（1）变量。

- ZERO 即常数 0，声明方式为

```
public static final BigInteger ZERO
```

- ONE 即常数 1，声明方式为

```
public static final BigInteger ONE
```

（2）（构造方法）构造方法 BigInteger 的使用方法。

- 一个 BigInteger 表示为正负号 signum（－1 表示负整数、0 表示 0、1 表示正整数）以及 byte 矩阵的大整数 magnitude 所组成的数。

```
public BigInteger(int signum,
byte[] magnitude)
                        throws NumberFormatException//例外处理
```

- 一个 BigInteger 表示为 radix-进位 val 字串的数。

```
public BigInteger(String val,int radix)
   throws NumberFormatException//例外处理
```

- 一个 BigInteger 表示为十进制 val 字串的数。

```
public BigInteger(String val)
```

注[7] 参阅http://java.sun.com/j2se/api/java/math/Biginteger.html。

```
throws NumberFormatException//例外处理
```

- 一个 0 至 $2^{numBits}-1$ 的随机 BigInteger 整数。

```
public BigInteger(int numBits,Random rnd)
throws IllegalArgumentException//例外处理
```

- 利用 Miller-Rabin 质数测试法找出位数为 bitLength、质数概率 $\geqslant 1-\left(\frac{1}{2}\right)^{certainty}$ 的随机可能质数。

```
public BigInteger(int,bitLength,
int certainty,Random rnd)
throws ArithmeticException//例外处理
```

（3）方法。

- valueOf

```
public static BigInteger valueOf(long val)
```

将类型为 long 的整数 val 化为 BigInteger。

- add

```
public BigInteger add(BigInteger val)
```

执行加法计算 this+val 并以 BigInteger 类型返回计算值。

- subtract

```
public BigInteger subtract(BigInteger val)
```

执行减法计算 this－val 并以 BigInteger 类型返回计算值、

- multiply

```
public BigInteger multiply(BigInteger val)
```

执行乘法计算 this*val 并以 BigInteger 类型返回计算值。

- divide

```
public BigInteger divide(BigInteger val)
throws ArithmeticException
//例外处理当 val=0
```

执行除法计算 this/val 并以 BigInteger 类型返回计算值。

- divideAndRemainder

```
public BigInteger[] divideAndRemainder(
BigInteger val)
throws ArithmeticException
//例外处理当 val=0
```

执行除法计算 this/val 并以 BigInteger 矩阵类型返回商数、余数。

- remainder

```
public BigInteger[] remainder(
BigInteger val)
        throws ArithmeticException
//例外处理当 val=0
```

执行除法计算 this/val 并以 BigInteger 矩阵类型返回余数。

- pow

```
public BigInteger pow(BigInteger exponent)
```

```
throws ArithmeticException
//当 exponent<0
```

执行指数计算 $this^{exponent}$ 并以 BigInteger 类型返回计算值。

- gcd

```
public BigInteger gcd(BigInteger val)
```

执行计算 $gcd(|this|, |val|)$ 并以 BigInteger 类型返回计算值。

- abs

```
public BigInteger abs()
```

执行计算绝对值 $|this|$ 并以 BigInteger 类型返回计算值。

- negate

```
public BigInteger negate()
```

执行变号计算 -this 并以 BigInteger 类型返回计算值。

- signum

```
public int signum()
```

取 this 的正负号（若 this＝0 则取 0）并以 int 类型返回计算值。

- mod

```
public BigInteger mod(BigInteger m)
throws ArithmeticException
// m<=0
```

执行电脑模计算 this%m 并以 BigInteger 类型返回计算值。

- modPow

```
public BigInteger modPow(BigInteger exponent,
BigInteger m)
throws ArithmeticException
// m<=0
```

执行指数模计算 $this^{exponent} \pmod m$ 并以 BigInteger 类型返回计算值。

- modInverse

```
public BigInteger modInverse(BigInteger m)
throws ArithmeticException
// this 与 m 不互质或 m<=0
```

执行乘法反元素计算并以 $this^{-1} \pmod m$ BigInteger 类型返回计算值。

- shiftLeft

```
public BigInteger shiftLeft(int n)
```

执行位向左计算 $this \ll n$（相当于 $this \times 2^n$）并以 BigInteger 类型返回计算值。

- shiftRight

```
public BigInteger shiftRight(int n)
```

执行位向右边计算 $this \gg n$（相当于 $this / 2^n$）并以 BigInteger 类型返回计算值。

- bitLength

```
public int bitLength()
```

计算 this 的位数（不含正负号）。

- isProbablePrime

```
public boolean isProbablePrime(
int certainty)
```

利用 Miller-Rabin 质数测试法测试 this 是否为质数，若为 false 则一定为非质数，若 true 则可能为质数，其概率为 $\geq 1-\left(\frac{1}{2}\right)^{certainty}$ 。

- compareTo

  ```
  public int compareTo(BigInteger val)
  ```

  比较 BigInteger 类型的 this 与 val，

$$\text{x.compareTo(y)} = \begin{cases} -1, & 若x < y \\ 0, & 若x = y \\ 1, & 若x > y。\end{cases}$$

- equals

  ```
  public boolean equals(BigInteger val)
  ```

  比较 this＝＝val。

- toString

  ```
  public String toString(int radix)
  ```

  将 BigInteger this 转化为 radix-进位的字符串。

- toString

  ```
  public String toString()
  ```

  将 BigInteger 转化为十进位的字符串。

程序 9.4（Jave / 强质数）：利用 Gordan 算法找出至少为 512 位的可能强质数。

输出≪强质数 p。

```
import java.math.BigInteger;
import java.security.*;//安全随机数
class strongPrime
{
    public static void main(String[] args)
    {
        SecureRandom rnd=new SecureRandom();
        int certainty=80;
        BigInteger
            s=new BigInteger(128,certainty,rnd);
        BigInteger
            t=new BigInteger(128,certainty,rnd);
        // s 与 t 为 128 位 的可能质数
        BigInteger r, ONE=BigInteger.valueOf(1);
        BigInteger i=ONE;
        static final BigInteger TWO=new BigInteger("2");
        do
        {
            r=TWO.multiply(i).multiply(t).add(ONE);
            //r=2*i*t+1
            i=i.add(ONE);
```

```
            //i++
        } while (!r.isProbablePrime(certainty));
        BigInteger p0=s.modPow(r.subtract(TWO),r).
            multiply(s).multiply(TWO).subtract(ONE);
        //p0=2*(s^(r-2)%r)*s-1;
        BigInteger j=ONE;
        do
        {
            p=TWO.multiply(j).multiply(r).multiply(s).
            add(p0);
                //p=p0+2*j*r*s
                j=j.add(ONE);
                //j++;
            } while (!p.isProbalePrime(certainty))
            int b=p.bitLength();
            System.out.println(b+"位的可能强质数 p="+p);
    }
}
```

9.8　大整数算术与数论套件及软件

对于初次学习密码学的读者，在网络资源中找到适当的套件或软件，再配合本身对程序语言的掌握，才能事半功倍，然而这些套件通常与计算数论有直接的关联，有时整个套件所涵盖的范围远远超过大整数计算或质数问题，即使要明白整个套件的内容，也需要相当了解其数学背景，这不是一两章的内容所能涵盖的；然而网络的资源很多，许多国外学术单位、研究机构早于多年前开始建构自己的程序库，并开放资源供外界共享，懂得利用这些资源，要比只是闭门造车，无论是对学习或研究都有更佳的立足点。

功能强大的数学软件，有 Maple 以及 Mathematica，这两个软件都有处理大整数计算以及处理质数问题的能力，甚至对于程序设计毫无概念的人也能马上上手，非常便利。Wade Trappe 与 Laerence C. Washington 所著的"Introduction to Cryptography with Coding Theory"一书，就附上大量要用 Maple 或 Mathematica 软件回答的习题，有兴趣的读者，可到 www.prenhall.com/washington，下载其 Maple 或 Mathematica 套件，而 Maple 及 Mathematica 的官方网址分别为 http://www.maplesoft.com 以及 http://www.mathematica.com，读者若无此软件，也可到官方网站下载试用版；部分以 Mathematica 软件所写的套件，也可在 http://www.mathematica.com 取得。

对于习惯自己动手写程序的读者，或要自行开发密码系统的读者，也有学习其他专家所写程序库的需求，以提高自身的能力。较为完整的大整数算术以及计算数论的软件套装表，可通过以下网址找到。

http://www.maths.uq.edu.au/~krm/web.html

http://www.math.uga.edu/~ntheory/web.html

http://www.dpmms.cam.ac.uk/Number-Theory-Web/web.html

下列软件套件只是这类软件套件中的一小部分。

- ARIBAS：为一个直译式的软件，可处理大整数的算术以及数论函数问题，ARIBAS 可在 ftp.mathematik.uni-muenchen.de 取得。
- CALC：由 Keith Matthews 开发，该套件以 ANSI C 撰写，可以在 ftp://www.maths.uq.edu.au/pub/krm/calc/取得。
- GNU MP 或 GMP：是一个 C 函数程序库，可处理任意整数的算术，也包括有理数、实数算术，该 C 程序库的函数拥有良好的执行能力，因为部分程序代码是用组合语言撰写的，在 www.gnu.org 以及 prep.ai.mit.edu 取得。
- LiDIA：为德国 Darmstadt 科技大学所开发的软件程序库，主要是处理数论上的计算问题，所以内容相当庞大，除了大整数、有理数、实数算术的函数程序，也包括不同代数结构上的计算，如 Galois 域 $\mathbb{F}_{p^n}$、椭圆曲线、代数体、多项式计算等。LiDIA 支持其他套件如 GNU 的界面，也支持不同版本 C++编译器，这在开发软件上要比 ANSI C 便利得多，甚至比 Java BigInteger Class 开发更符合语言逻辑一致性（可惜的是 Java 不支持运算符重载）。这可在 http://www.informatik. tu-darmstadt.de/TI/LiDIA 取得。
- PARI：由 Henri Cohen 等人开发的数论套件，PARI 本身可直接拿来使用或当作函数程序库连接到其他程序使用，并使用不同平台上的组合语言撰写，执行能力强，（可在 UNIX、Machintosh、PC 上执行），PARI 可在 www.parigp-home.de 取得。

第 10 章　椭圆曲线密码

椭圆曲线（Elliptic Curve）的研究至少已经有 150 年的历史，一般认为它是数论以及代数几何所研究的课题。有关椭圆曲线的课题，足以让不少数学家无休止地讨论下去，过去数学家根本不在乎椭圆曲线的实用价值，关心的是其丰富的性质，早期根本没人知道椭圆曲线会有何实用价值。

但自从 Miller 以及 Koblitz 注[1]分别发表如何将椭圆曲线引入密码学的文章，椭圆曲线成了“跨领域”的热门话题；数百年未解的最后费马定理也由 Andrew Wiles 证明，他的方法就是以椭圆曲线为研究基础的。椭圆曲线密码系统虽然尚未成为市场的主流，而且许多细节尚有改善的余地，甚至也有专家怀有疑虑，“不似 RSA，因数分解问题早已被人类研究得相当透彻，其安全性已被证实；而人类对椭圆曲线根本还不太了解，以椭圆曲线为架构的密码系统，很可能有快速之道，只是还未被发现”。这类的质问，当然不无道理。尽管如此，密码界还是逐渐接受了椭圆曲线，RSA 公司的 PKCS#14 就准备定义椭圆曲线密码系统（Elliptic Curve Cryptosystem，ECC)，IEEE 的 P1363 以及 FIPS 也定义了椭圆曲线公开密钥密码。大力推动的还有加拿大一家密码公司 Certicom，公开在网站提出椭圆曲线离散对数问题（Elliptic Curve Discrete Log Problem，ECDLP）的悬赏。

椭圆曲线之所以在密码学中受到重视，因为除了特定曲线外，相信是不存在指数次时间的有效攻击法，它的安全性远高于以简易模运算为主（包括以因数分解与离散对数为基础）的各种公开密钥密码系统：一般有效的攻击法为并行 Pollard Rho 法，注[2]其计算复杂度为 $O(2^{n/2})$，其中椭圆曲线密码的密钥为 n 位。以目前所知破译算法评估计算复杂度，密钥为 1024 位的 RSA 密码，其安全性大约等同于密钥为 163 位的椭圆曲线密码，可详见表 10-1，如图 10-1 所示。

表 10-1　NIST 所估计不同长度密钥的 RSA 与椭圆曲线密码系统所提供的相同的计算安全度

椭圆曲线密码	112 位	163 位	224 位
RSA	512 位	1024 位	2048 位
密钥长度比	5:1	6:1	9:1
椭圆曲线密码	256 位	384 位	512 位
RSA	3072 位	7680 位	15360 位
密钥长度比	12:1	20:1	30:1

注[1] N.Koblitz 的“Elliptic Curve cryptosystem.” Mathematics of Computation 48, P.203-209, (1987).Miller 的“Use of elliptic Curve in cryptography” in “Advances in Cryptology, CRYPTO 85” Springer Verlag,LNCS210, P.417(1986)。

注[2] Paul C.van Oorscht 与 Michael J.Wiener 的“Parrallel collision search with cryptanalytic applications.” Journal of Cryptography,12P .1-28,(1999)。

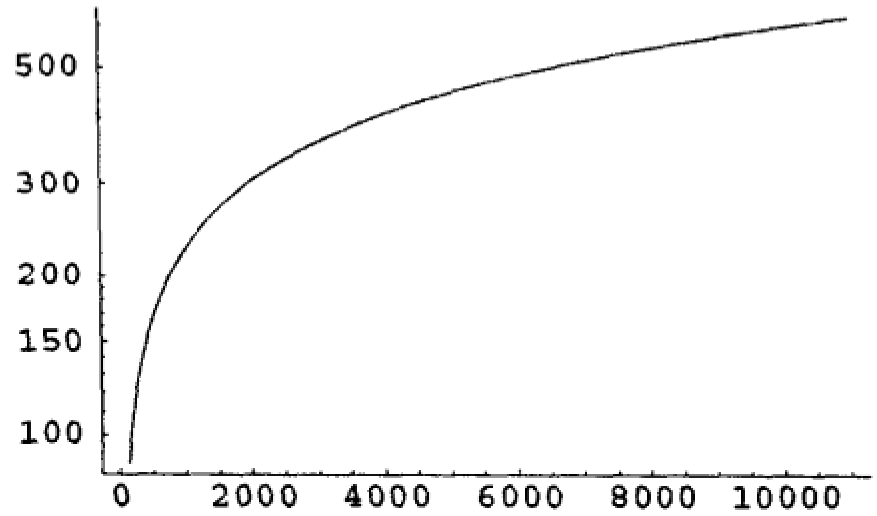


图 10-1 相同的计算安全度的 RSA 密钥位（y-轴）与椭圆曲线密码密钥位（x-轴）（RSA 以代数域筛法估计，椭圆曲线密码以 Pollard Rho 法估计）

而椭圆曲线上的“乘法”运算比同位数的模指数运算慢，但还属于同数量级，因此，在要求相同安全性的情形下，椭圆曲线密码的加密解密速度要比 RSA 密码快上许多；同时，短密钥的椭圆曲线密码可达到与长密钥RSA密钥相同的计算安全度，也非常适用于在IC卡上使用，其实用价值不言而喻。

10.1 椭圆曲线

本节首先介绍椭圆曲线的基本概念及性质。

定义 10.1 椭圆曲线即三次平滑代数平面曲线（Smooth Algebraic Plane Curve），可在适当的坐标下，表示成 Weierstrass 方程式

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

除了特殊系数K，[注[3]]一般而言，可在适当的坐标下，表示成

$$E: y^2 = x^3 + ax + b \text{（判别式}\Delta = -16(4a^3 + 27b^2) \neq 0\text{）}$$

在系数 K 上的平面图形，其中 E 也包括无限远点 $\mathcal{O}$。[注[4]]

注 10.1

（1）系数 K 的选取可为 $\mathbb{R}$、$\mathbb{C}$、$\mathbb{Q}$ 及 $\mathbb{Z}/p$（p 为质数）等，在椭圆曲线密码中常用 $K = \mathbb{Z}/p$。

（2）若判别式 $\Delta = -16(4a^3 + 27b^2) = 0$，此时 $E: y^2 = x^3 + ax + b$ 不再是平滑曲线。

（3）当 $K = \mathbb{R}$，可利用微积分技巧描绘示意图，其图形可分为两类，如图 10-2 所示。

1）$x^3 + ax + b = 0$ 有3个相异实根，则图形有两个分量(Component)，如 $y^2 = x(x-3)(x+4)$。

2）$x^3 + ax + b = 0$ 有一个实根及两个共轭复根，则图形为连通的（Connected），如 $y^2 = x^3 + 13$。

（4）椭圆曲线不是椭圆，其名源自计算椭圆弧长的椭圆积分（Elliptic Integral）如

$$\int_{z_1}^{z_2} \frac{dx}{\sqrt{x^3 + ax + b}} \text{ 及 } \int_{z_1}^{z_2} \frac{xdx}{\sqrt{x^3 + ax + b}}\text{。}$$

注[3] 特殊系数指 K 的特征数（Characteristic）为 2 或 3。

注[4] 采用投影几何（Projective Geometry）的公平公设：平面上的两根平行线相交于无限远点。

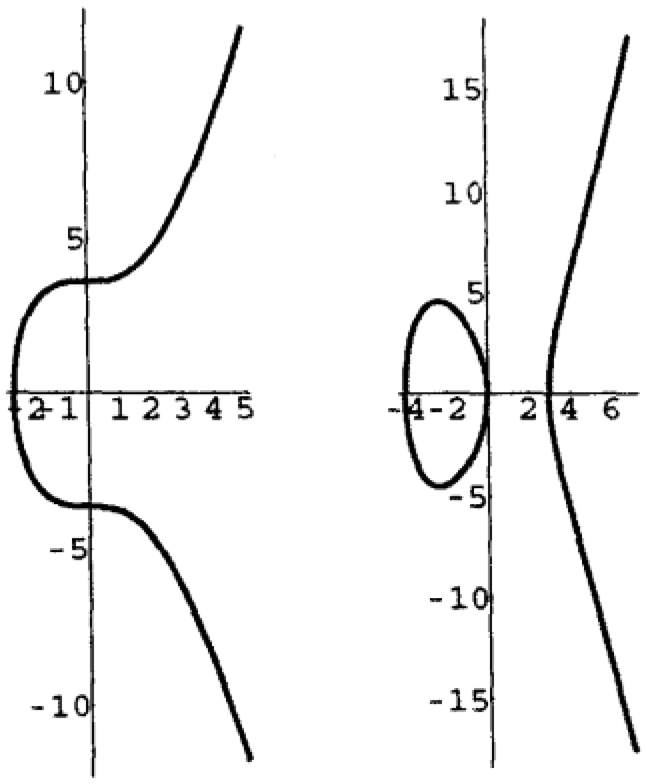


图 10-2　$y^2 = x^3 + 13$ 与 $y^2 = x(x-3)(x+4)$

椭圆曲线密码主要用到椭圆曲线的代数群（Algebraic Group）结构，而其中的运算"加法"可用截线、切线等几何概念构建。

定义 10.2（椭圆曲线加法律）　在椭圆曲线 $E: y^2 = x^3 + ax + b$ 定义"加法"如下：

（1）视无限远点 $\mathcal{O}$ 为"加法单位素"。

（2）点 P 的"加法反元素"即点 $-P$，定义为点 P 对 x 轴的镜像。

（3）一般而言，三次曲线与直线相交于三点（需计算重数（Multiplicity），若相切时，重数为 2，如图 10-3 所示，P、Q、$-(P+Q)$ 共线，点 $P+Q$ 即点 $-(P+Q)$ 对 x 轴的镜像，依此定义"加法"。

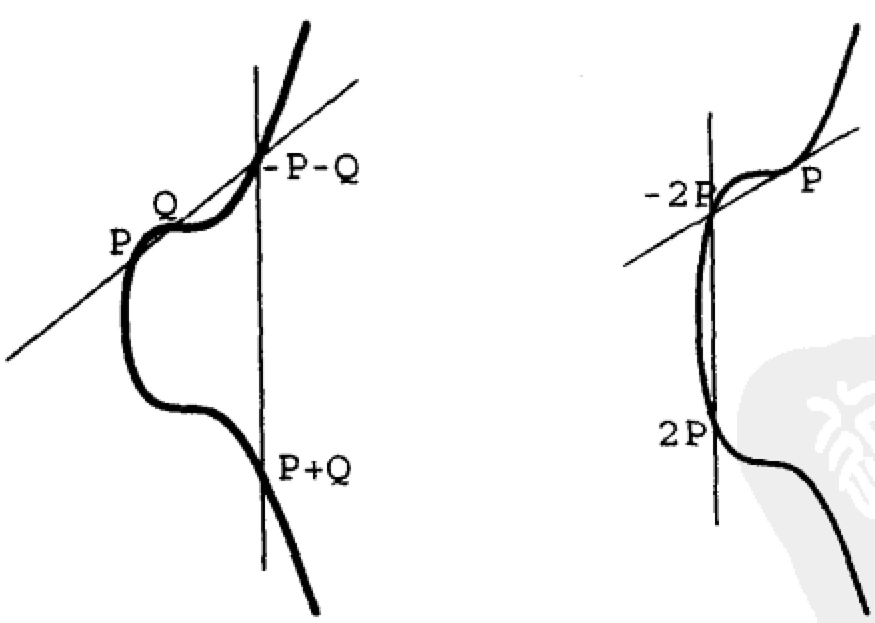


图 10-3　椭圆曲线加法律

（4）加法坐标计算：令 $P = (x(P), y(P))$、$Q = (x(Q), y(Q))$，欲求 $R = P + Q = (x(R), y(R))$，其中可分为三种情形：

- $x(P) \neq x(Q)$：取通过 P、Q 截线的斜率 $m = \dfrac{y(Q)-y(P)}{x(Q)-x(P)}$。

- $x(P)=x(Q)$、$y(P)=y(Q)$：即 $P=Q$，取通过 P 切线的斜率 $m=\dfrac{3x(P)^2+a}{2y(P)}$。
- $x(P)=x(Q)$、$y(P)=-y(Q)$：此时 $R=P+Q=\mathcal{O}$。

除第 3 种情形外，

$$R=(x(R),y(R))=(m^2-x(P)-x(Q),m(x(P)-x(R))-y(P))。$$

定理 10.1 $(E,+,\mathcal{O})$ 为交换群（Abelian Group），其中加法"+"，如上所定，无限远点 $\mathcal{O}$ 为加法单位素。

注意：证明需检查封闭性、交换律、加法单位素、加法反元素以及结合律，除结合律外，几何方式较为直接，但若不用坐标计算，证明结合律 $(P_1+P_2)+P_3=P_1+(P_2+P_3)$ 较为困难，需用到代数几何。而此处"加法"计算 P_1+P_2 需若干加减乘除，其计算复杂度与乘除大致相当。

有了加法律，自然可在椭圆曲线上定义"乘法"，而乘法运算是椭圆曲线密码系统的重要运算。

定义 10.3 令 P 为椭圆曲线 E 上一点。对自然数 n，可定义"乘法"

$$[n]P:=\underbrace{P+\cdots+P}_{n个}$$

而计算[n]P 的方式，类似计算指数的二元法计算，而加、乘法的算法，可用 C++ Class 算符重载的方式呈现。

程序 10.1 在椭圆曲线 $E/\mathbb{Q}:y^2=x^3+ax+b$ 上计算 $P+Q$、$[n]P$：

```
friend point operator+(point& P,point& Q)
{
    point R=O_Inf;//设为无穷远点
    bigrational m;// 有理数
    if (P==O_Inf)//加法单位素
        return Q;
    else if (Q==O_Inf)//加法单位素
        return P;
    else if (P.x==Q.x && P.y==Q.y==0)//Q=P
        return R;//P+P=O_Inf
    else if (P.x==Q.x && P.y!=Q.y)//Q=-P
        return R;//P+(-P)=O_Inf
    else if (P==Q)
        m=(3*P.x*P.x+a)/(2*P.y);//切线斜率
    else
        m=(Q.y-P.y)/(Q.x-P.x);//截线斜率
    R.x=m*m-P.x-Q.x;
    R.y=m*(P.x-R.x)-P.y;
    return R;
};
friend point operator*(biginteger n,point& P)
{
    point result=O_Inf;//设为无穷远点
    if (n<0)
    {
```

```
        n=-n;
        P.y=-P.y;
        //[n]P=[-n](-P)
    };
    while (n!=0)
    {
        if (is_odd(n))
            result=result+P;
            P=P+P;
        n>>=1;
    };
    return result;
};
```

注 10.2　上述“乘法”算法，可再进一步改进。若 $P=(x,y)$，计算 $[1023]P$ 时，就可考虑计算

$$[1023]P=[2^{10}-1]P=\overbrace{[2]([2](\cdots([2]P)\cdots))}^{10次2倍}+(-P)$$
$$=\overbrace{[2]([2](\cdots([2](x,y))\cdots))}^{10次2倍}+(x,-y)$$

其中

$$1023=(111111111)_2=(1000000000)_2-(1)_2=(1,0,0,0,0,0,0,0,0,-1)_2,$$

原先要花费“两倍”9 次、“加法”9 次，利用上式计算只要花费“两倍”10 次，“加法”2 次，计算上更为简便。任何整数 k 都可先化为二进制，再化为所谓 Non-Adjacent 形态，即相邻两位必至少有一位是 0，与二进制不同，其各位数可用 −1，0，1，这样在“乘法”计算中平均可节省 11%的计算量。如 $k=23$ 时，其 Non-Adjacent 形态为：

$$23=(0,\ 1,\ 0,\ 1,\ 1,\ 1)_2$$
$$=(0,\ 1,\ 1,\ 0,\ 0,\ -1)_2$$
$$=(1,\ 0,\ -1,\ 0,\ 0,\ -1)_2\ 。$$

例 10.1　椭圆曲线 $E/\mathbb{Q}: y^2=x(x-3)(x+4)$ 中的点

$$\mathcal{O}、(0,0)、(0,3)、(0,-4)$$

满足

$$(0,-4)+(0,0)=(0,3),$$
$$(0,0)+(0,3)=(0,-4),$$
$$(0,3)+(0,-4)=(0,0),$$
$$[2](0,-4)=\mathcal{O},$$
$$[2](0,0)=\mathcal{O},$$
$$[2](0,3)=\mathcal{O},$$

这些都可直接以几何方式验证。

例 10.2　椭圆曲线 $E/\mathbb{Q}: y^2=x^3+1$ 中的点

$$\mathcal{O}、(0,1)、(0,-1)$$

满足

$$\begin{aligned}(0,1)+(0,-1) &= \mathcal{O}\ , \\ [2](0,1) &= (0,-1)\ , \\ [2](0,-1) &= (0,1)\ , \\ [3](0,1) &= \mathcal{O}\ , \\ [3](0,-1) &= \mathcal{O}\ 。\end{aligned}$$

例 10.3 在椭圆曲线 $E/\mathbb{Q}: y^2 = x^3 + 15$ 上计算点 $P=(1,-4)$ 的“椭圆曲线乘法”：

$P=(1,-4)$

$[2]P = (-\frac{119}{64}, \frac{1499}{512})$

$[3]P = (\frac{225361}{33489}, \frac{109585196}{6128487})$

$[4]P = (\frac{3943950241}{575232256}, -\frac{253381593538319}{13796370427904})$

$[5]P = (-\frac{623089305949199}{338865324807025}, -\frac{18486988064660195488076}{6237932864318534272375})$

$[6]P = (\frac{1563675758466438261961}{1608666242168003973696}, \frac{25742393748541628678148263577306 1}{64520677993442490021369331616256})$

$[7]P = (\frac{4874891678033220784974900984 01}{597329816750517970660348 9},$
$\frac{340367006375296786342221236524744876280996644}{1459893889410910246131061620631037 4737})$

$[8]P = (\frac{151867143945809200380627875652764595841}{147724778885570731158765033519424267264},$
$-\frac{72012993091213226484527917613256816589863807492168903254 41}{17954777305217112319457974921032166941013818399440811130 88})$

$[9]P = (-\frac{53998438894992783487689679851869599611289513963 19}{28726488802502218748889167928557998400604928289 61},$
14075899363740736325521199066521586727220945593 83
6221063632678871395321116 / 4868821121834674288885 9
874955478989215090294745936696067836967762094682 09)

由上例可知，定义在有理数上的椭圆曲线，其实只要由“很小”的点 P，注[5]不需要很大的整数 m，就可得到“很大”的点[m]P，这与一般整数的加法、乘法不同。然而椭圆曲线上的运算并非无迹可寻，其实数论家对于定义在有理数 $\mathbb{Q}$ 的椭圆曲线 E 的代数结构也有相当深刻的认识，这在密码学应用上有一定的帮助。其中之一认识较为深刻：

定理 10.2（Mordell-Weil） 令椭圆曲线 E 定义在有理数 $\mathbb{Q}$ 上，$E(\mathbb{Q})$ 为曲线上所有 x、y 坐标皆为有理数的点所成的集合（也包含无穷远点 $\mathcal{O}$），则

注[5] 较为精确的说法，就要用刀 Height 函数。可参阅 Josef H.Silverman 的“The Arithmetic of Elliptic Curves”，Springer-Verlag（1985）。

$$E(\mathbb{Q}) \cong \mathbb{Z}^r \times T,$$

其中 r 为某正整数，而

$$T = \{P \in E(\mathbb{Q}) \mid 存在某正整数 n 使得 [n]P = \mathcal{O}\}$$

为一有限交换群，一般称之为 Torsion 子群。

注意：r 为自然数，但可能很大，有人猜测 r 可任意大，迄今无人会证或反证；而数论家 Mazur 证明 T 只有几种可能。这些都是相当深刻的描述，需要不少理论。而例 10.1 与例 10.2 的各点均为 Torsion 点。

10.2　椭圆曲线（mod p）

在此，我们的质数 $p>3$，在前一节中我们介绍了椭圆曲线，一般而言，密码学中所用的椭圆曲线最常见的有两类：

- 定义在 Galois 域 $\mathbb{F}_p = \mathbb{Z}/p$（p 为大质数）上的椭圆曲线，即质数曲线（Prime Curve）。
- 定义在 Galois 域 $\mathbb{F}_{2^m}$（m 为大整数）的椭圆曲线，即二元曲线（Binary Curve）。

两者都是代数群，其中定义在 $\mathbb{F}_p$ 上的椭圆曲线，可直接由定义在 $\mathbb{Z}$ 上的椭圆取模运算 (mod p) 取得，其群结构也承袭前者，我们将其整理为如下定理。

定理 10.3　令 $p>3$ 为质数。令定义在整数 $\mathbb{Z}$ 上的椭圆曲线

$$E: y^2 = x^3 + ax + b,$$

其中系数 a、$b \in \mathbb{Z}$ 均为整数且满足

$$4a^3 + 27b^2 \not\equiv 0 \pmod p。$$

则

$$\tilde{E}: y^2 \equiv x^3 + ax + b \pmod p$$

为定义在 $\mathbb{F}_p$ 上的椭圆曲线。

而且椭圆曲线 $\tilde{E}$ 的“加法”运算继承自椭圆曲线 E，[注6]详言之，若 P、$Q \in E(\mathbb{Z})$，则 $\tilde{P} = P \pmod p$、$\tilde{Q} = Q \pmod p$ 均在 $\tilde{E}(\mathbb{F}_p)$ 上，其“加法”可定义为

$$\tilde{P} + \tilde{Q} = \widetilde{P+Q}$$

即 P、Q 两点在 $E(\mathbb{Z})$ 上先“加法”再取模运算 (mod p)。

注意：模运算不只是对数字，也可对定义在整数上的椭圆曲线 E，而做出定义在 $\mathbb{F}_p = \mathbb{Z}/p$ 上的椭圆曲线 $\tilde{E}$，其“加法”也是先在“楼上”E 做好，结果坐标经模运算再搬到“楼下”$\tilde{E}$。但若等式

$$4a^3 + 27b^2 \equiv 0 \pmod p$$

成立，造出来的 $\tilde{E}$ 就不是椭圆曲线。

例 10.4　令椭圆曲线 $E: y^2 = x^3 + x + 1 \pmod 5$。如何找出 E 上的所有的 $\mathbb{F}_5$ 点？将 0、1、

注[6] 以代数之术语而言，就是存在一个 Homomorphism $\phi:(E,+,\mathcal{O}) \to (\tilde{E},+,\mathcal{O})$，即

$$\phi(\mathcal{O})=\mathcal{O} 且 \phi(P)+\phi(Q)=\phi(P+Q)$$

2、3、4 代入 $f(x)=x^3+x+1$ 分别得

$$f(0)=1、f(1)=3、f(2)=1、f(3)=1、f(4)=4 \quad (\text{mod } 5)$$

值 $z=1$，4 在 $\mathbb{F}_5$ 有二次剩余（有平方根），

$$因为\left(\frac{z}{5}\right)\equiv z^{\frac{5-1}{2}}\equiv\begin{cases}1 & (\text{mod } 5) \quad 若z=1,4\\ -1 & (\text{mod } 5) \quad 若z=2,3\end{cases}$$

故

$$E(\mathbb{F}_5)=\{\mathcal{O},(0,\pm1),(2,\pm1),(3,\pm1),(4,\pm1)\}$$

共有 9 点（不要忘了也包括无穷远点）。此时 $P=(0,1)$，代入加法坐标公式计算 $[n]P$ 可得

$$\begin{aligned}
[2]P &= (4,2)\\
[3]P &= (2,1)\\
[4]P &= (3,4)\\
[5]P &= (3,1)\\
[6]P &= (2,4)\\
[7]P &= (4,3)\\
[8]P &= (0,4)\\
[9]P &= \mathcal{O}
\end{aligned}$$

我们发现所有在 $E(\mathbb{F}_5)$ 的点均是由点 $P=(0,1)$ 所“生成”的，故 $E(\mathbb{F}_5)$ 为一秩为 9 的循环群，此时称 P 为代数群 $E(\mathbb{F}_5)$ 的生成元。

而在密码学上的应用就是要考虑椭圆曲线离散对数问题（Elliptic Curve Discrete Log Problem，ECDLP）。

定义 10.4（椭圆曲线离散对数问题） 令 $E: y^2=x^3+ax+b \pmod p$ 为一椭圆曲线，令 P、Q 为 $E(\mathbb{F}_p)$ 上的两点，假设点 Q 由点 P 生成。在 E 上的离散对数问题就是要解

$$Q=[k]P$$

的 k 值。

例 10.5 令椭圆曲线 $E: y^2=x^3+x+1 \pmod 5$。令 $P=(4,2)$，$Q=(4,3)$。求

$$Q=[n]P$$

的 n。

由例 10.3 知

$$\begin{aligned}
P&=(4,2)=[2](0,1),\\
Q&=(4,3)=[7](0,1),\\
E(\mathbb{F}_5)&=9,
\end{aligned}$$

故

$$Q=(4,3)=[n]P=[n]([2](0,1))=[2n](0,1)=[7](0,1),$$

所以

$$2n\equiv 7 \quad (\text{mod } E(\mathbb{F}_5)=9),$$

解得

$$n\equiv 8 \quad (\text{mod } 9)。$$

上个世纪，数论学家对椭圆曲线，特别是定义在 Galois 域上的椭圆曲线都有相当程度的认识，其中对 $E(\mathbb{F}_p)$ 的个数的估计也有相当程度的理解。

定理 10.4　令 E 为定义在 $\mathbb{F}_p$ 上的椭圆曲线，则 $E(\mathbb{F}_p)$ 的个数满足

$$\#E(\mathbb{F}_p) = p + 1 + t$$

其中误差项

$$|t| < 2\sqrt{p}\text{。}$$

实际上，这与 Frobenius 转换

$$\mathrm{Frob}: E \to E$$

$$\mathrm{Frob}: \begin{cases} (x,y) \mapsto (x^p, y^p) \\ \mathcal{O} \mapsto \mathcal{O}\text{。} \end{cases}$$

有密切关联：

$$(x^{p^2}, y^{p^2}) - [t](x^p, y^p) + [p](x,y) = \mathcal{O}\text{。}$$

其实 Frobenius 转换也为线性变换，因此依据线性代数原理，可视为一个 2×2 的矩阵，而此时 $t = \mathrm{tr}(\mathrm{Frob})$，即矩阵的迹（Trace）。

在实际应用中，给出一条椭圆曲线

$$E(p,a,b): y^2 \equiv x^3 + ax + b \pmod{p},$$

要计算出 $\#E(p;a,b)/\mathbb{F}_p$ 虽不能显而易见，但也有合理时间内可算出的算法，注[7]在 $\ln p$ 的多项式时间内可完成。但若 $p < 10000$ 很小时，也可直接利用公式：

定理 10.5　令椭圆曲线 $E: y^2 = x^3 + ax + b \pmod{p}$，则

$$g = \#E(\mathbb{F}_p) = p + 1 + \sum_{x=0}^{p-1}\left(\frac{x^3 + ax + b}{p}\right),$$

其中 $\left(\frac{\cdot}{p}\right)$ 为 Legendre 符号。

证明：令 $f(x) = x^3 + ax + b$，考虑 Legendre 符号 $\left(\frac{x^3 + ax + b}{p}\right)$（$x = 0, 1, \cdots, p-1$）：

- $\left(\frac{x_0^3 + ax_0 + b}{p}\right) = 1$：此时 $(x_0, \pm\sqrt{f(x_0)})$ 两点均为 $E(\mathbb{F}_p)$ 上的点。
- $\left(\frac{x_0^3 + ax_0 + b}{p}\right) = -1$：此时 $y^2 \equiv f(x_0) \pmod{p}$ 在 $\mathbb{Z}/p = \mathbb{F}_p$ 无解，故 $E(\mathbb{F}_p)$ 上无 x 坐标点为 x_0 的点。

注[7] 其实者有相当的难度，以作者所知约分为两大类算法处理指数曲线：

1．嫌疑随机方式产生曲线，以 Shoof 算法以及其推广数点。

2．先驱定点的个数，再以所谓的 CM 发构建曲线。

前者计算虽能在多项式时间内完成，但仍很复杂，方法较为直接；而后者需要用到大量的椭圆曲线，较为迂回，但计算容易。

- $\left(\frac{x_0^3+ax_0+b}{p}\right)=0$：此时 $(x_0,0)$ 为 $E(\mathbb{F}_p)$ 上的点。

再加上无穷远点 $\mathcal{O}$ 一点，可得公式。

这样的方式计算需要 $\ln p$ 的指数时间才能完成。但若已知某点 P 的秩 ord(P)，根据群论的 Lagrange 定理，必然

$$\text{ord}(P)\mid g=\#E(\mathbb{F}_p)\text{ 且 }[g]P=\mathcal{O}。$$

这对 $g=\#E(\mathbb{F}_p)$ 的个数的估计也有相当帮助。

例 10.5 令椭圆曲线

$$E: y^2=x^3-x+1 \pmod{19},$$

其中

$$(0,13)、P=(1,1)\in E(\mathbb{F}_{19}),$$

并且

$$[2](0,13)=\mathcal{O},$$

故

$$\text{ord}(0,13)=2\mid \#E(\mathbb{F}_{19});$$

另外

$$\begin{aligned}[2]P &= (18,1)\\ [3]P &= (0,18)\\ [4]P &= (3,14)\\ [5]P &= (5,11)\\ [6]P &= (5,8) = -[5]P,\end{aligned}$$

故

$$[11]P=\mathcal{O}$$

所以

$$\text{ord}(P)=11\mid \#E(\mathbb{F}_{19});$$

因此

$$22\mid \#E(\mathbb{F}_{19}),$$

但由 Hasse 定理可得

$$12=\left\lceil 19+1-2\sqrt{19}\right\rceil\leqslant \#E(\mathbb{F}_{19})\leqslant\left[19+1+2\sqrt{19}\right]=28,$$

故

$$\#E(\mathbb{F}_{19})=22。$$

计算

$$R=(10,12)=(1,1)+(0,13),$$

其中

$$[n]R\quad(\gcd(n,22)=1)$$

的点均为 $E(\mathbb{F}_{19})$ 的生成元。

10.3　加权投影坐标

在椭圆曲线的加法公式中，需要作除法计算，这在密码学上的应用，意味着要计算模除法就必须使用广义辗转相除法，计算的成本较大。因此在程序编写时，就要考虑较为精细的算法。以 IEEE P1363 草案标准注[8]为例，就建议采用加权投影坐标（Weighted Projective Coordinate）来计算。以定义在 $\mathbb{Z}$ 上的椭圆曲线

$$E: y^2 = x^3 + ax + b$$

而言，就是要将整条曲线 E 嵌入适当的加权投影空间（Weighted Projective Space），即

$$P(2,3,1)$$ 注[9]

其实际做法就是除了 x 坐标、y 坐标外，再加上 z 坐标，并设定其次数

$$\deg(x) = 2, \deg(y) = 3, \deg(z) = 1,$$

（这就是“加权”的意义）其曲线方程式可改写成加权齐次（Weighted Homogeneous）的形式：

$$y^2 = x^3 + ax + b \to y^2 = x^3 + axz^4 + bz^6$$

平面坐标转换为投影坐标公式为

$$\begin{cases} (x,y) \mapsto [x,y,1] \\ \mathcal{O} \quad \mapsto [0,1,0] \end{cases}。$$

而投影坐标转换成平面坐标公式为

$$\begin{cases} [x,y,z] \mapsto \left(\dfrac{x}{z^2}, \dfrac{y}{z^3}\right) \quad 若 z \neq 0 \\ [0,y,0] \mapsto \mathcal{O} \end{cases}。$$

加权投影坐标椭圆加法公式可整理如下。

性质 10.1　令椭圆曲线

$$E: y^2 = x^3 + ax + b。$$

令 $P_1 = [x_1, y_1, z_1]$，$P_2 = [x_2, y_2, z_2]$ 为 E 上的点（加权投影坐标表达式）。而点

$$P_3 = [x_3, y_3, z_3] = P_1 + P_2。$$

若 P_1、$P_2 \neq \mathcal{O}$，且 $P_1 \neq \pm P_2$，则加法公式为

$$\begin{aligned} m_1 &= x_1 z_2^2 \\ m_2 &= x_2 z_1^2 \\ m_3 &= m_1 - m_2 \\ m_4 &= y_1 z_2^3 \\ m_5 &= y_2 z_1^3 \end{aligned}$$

注[8]　IEEE P1363/D3（Draft Version 3）。Shandard specifications for public key cryptography.May 1998。

注[9]　以复数加权投影空间就是

$$P(2,3,1) = \{[x,y,z] \mid (x,y,z) \in C^3, (x,y,z) \neq (0,0,0), (x,y,z) \sim (k^2x, k^3y, kz), k \in C^x\},$$

其中 $[x,y,z]$ 为等价类，而无限远点可表示为 O=[0，1，0]。

$$
\begin{aligned}
m_6 &= m_4 - m_5 \\
m_7 &= m_1 + m_2 \\
m_8 &= m_4 + m_5 \\
z_3 &= z_1 z_2 m_3 \\
x_3 &= m_6^2 - m_7 m_3^2 \\
m_9 &= m_7 m_3^2 - 2x_3 \\
y_3 &= (m_9 m_6 - m_8 m_3^3)/2
\end{aligned}
$$

- 若 $P_1 = P_2$，则 $[2]P_1 = [x_3, y_3, z_3]$ 的公式为

$$
\begin{aligned}
m_1 &= 3x_1^2 + az_1^4 \\
z_3 &= 2y_1 z_1 \\
m_2 &= 4x_1 y_1^2 \\
x_3 &= m_1^2 - 2m_2 \\
m_3 &= 8y_1^4 \\
y_3 &= m_1(m_2 - x_3) - m_3
\end{aligned}
$$

注意：以质数曲线

$$E: y^2 = x^3 + ax + b \pmod p$$

为例，加法公式中，还要计算 $2^{-1} \pmod p$，不过这是肯定都要用到的，可事先计算好储存起来，如此一来在实际中，就无须考虑模除法（即广义辗转相除法），反而要多计算多个模乘法，但整个计算过程还是要比计算一次模除法方例。另外，以加权投影坐标计算，其实只需要在整数上考虑，而任何平面有理数坐标，都可转换成加权投影整数坐标值，

$$(\alpha, \beta) = \left(\frac{x}{z^2}, \frac{y}{z^3}\right),$$

其中 $[x, y, z]$ 为 (α, β) 的加权投影坐标值，若坐标出现分数，也可转换成另一个整数坐标值，如

$$[x, y, z] = [k^2 x, k^3 y, kz]$$

其中 k 为适当整数。

例 10.6 令椭圆曲线 $E: y^2 = x^3 + x + 1 \pmod 5$。由例 10.4 已知共有 9 点（不要忘了也包含无穷远点），且 $P = (0,1)$ 生成 $E(\mathbb{F}_5)$，代入加权投影坐标加法坐标公式计算 $[n]P$ 可得

$$
\begin{aligned}
P &= (0,1) = [0,1,1] \\
[2]P &= [1,1,2] = (1 \times 2^{-2} \bmod 5,\ 1 \times 2^{-3} \bmod 5) = (4,2) \\
[3]P &= [3,2,3] = (3 \times 3^{-2} \bmod 5,\ 2 \times 3^{-3} \bmod 5) = (2,1) \\
[4]P &= [3,1,4] = (3 \times 4^{-2} \bmod 5,\ 1 \times 4^{-3} \bmod 5) = (3,4) \\
[5]P &= [2,2,3] = (2 \times 3^{-2} \bmod 5,\ 2 \times 3^{-3} \bmod 5) = (3,1) \\
[6]P &= [3,2,2] = (3 \times 2^{-2} \bmod 5,\ 2 \times 2^{-3} \bmod 5) = (2,4) \\
[7]P &= [1,4,2] = (1 \times 2^{-2} \bmod 5,\ 4 \times 2^{-3} \bmod 5) = (4,3) \\
[8]P &= [0,3,3] = (0 \times 3^{-2} \bmod 5,\ 3 \times 3^{-3} \bmod 5) = (0,4) \\
[9]P &= [0,1,0] = \mathcal{O}。
\end{aligned}
$$

例 10.7　在椭圆曲线 $E/\mathbb{Z}: y^2 = x^3 + 15$ 上以加权投影坐标计算点 $P = (1,-4)$ 的“乘法”：

$(1,-4) = [1,-4,1]$

$[2](1,-4) = [-119,-1499,8]$

$[3](1,-4) = [14423104,-56107620352,-1464]$

$[4](1,-4) = [3943950241,-253381593538319,23984]$

$[5](1,-4) = [-12003163117807513780874831616,$
$15630910505279555981624482087052539207188 48,$
$-80795332152240]$

$[6](1,-4) = [36842567032633769234945843774 09536,$
$93100860992609458381730987917808274411510671448 4736,$
$61565096368398336]$

$[7](1,-4) = [2461786027301084933595734942535665424145261264896,$
$3862561910902861451326686063535450860638416395738$
$6084759614826470556 30336,$
$5492243241857607770112]$

$[8](1,-4) = [15186714394580920038062787565276459 5841,$
$72012993091213226484527917613256816589863807492168903 25441,$
$-1215420827884608579 2]$

可比较例 10.3。

10.4　定义在 Galois 域 $\mathbb{F}_{2^m}$ 的椭圆曲线

值得一提的是，本节对于不太了解 Galois 域的读者可能有相当的难度，因此需要对大学代数有相当程度的了解。对于 Galois 域的理论可参阅本书的“基础数论”部分。注意 Galois 域 $\mathbb{F}_{2^m}$ 的特征值（Characteristic）为 2，故定义在 $\mathbb{F}_{2^m}$ 的椭圆曲线的 Weierstrass 方程式与先前稍有不同，整理如下：

定义 10.6　定义在特征值为 2 的 Non-Supersingular 注[10]椭圆曲线，可在适当的坐标系选取下表示成

$$E: y^2 + xy = x^3 + ax^2 + b$$

其中 a、$b \in K$ 且 $b \neq 0$，也包括无穷远点 $\mathcal{O}$。而其“加法律”可定义如下：令

$P = (x(P), y(P))$、$Q = (x(Q), y(Q))$，欲求 $R = P + Q = (x(R), y(R))$，假设 $P, Q \neq \mathcal{O}$ 则可分为 3 种情形：

- $x(P) \neq x(Q)$：

$$m = \frac{y(P) + y(Q)}{x(P) + x(Q)}$$

注[10] 特征值 p>0 的椭圆曲线有 Supersingular 与 Non-Supersingular 之分，密码学上是不用不安全的 Supersingular 椭圆曲线。

$$x(R) = m^2 + m + x(P) + x(Q) + a$$
$$y(R) = m(x(P) + x(R)) + x(R) + y(P)$$

- P=Q：

$$m = \frac{y(P)}{x(P)} + x(P)$$
$$x(R) = m^2 + m + a$$
$$y(R) = m(x(P) + x(R)) + x(R) + y(P)$$
$$x(P) = x(Q) \text{、} y(P) = -y(Q)\text{：} R = P + Q = \mathcal{O}\text{。}$$

在实际计算上以 $K = \mathbb{F}_{2^n}$ 为例，椭圆曲线的"加法"必须计算 $(x(P) + x(Q))^{-1}$ 以及 $(x(P))^{-1}$，即 $x(P) + x(Q)$ 及 $x(P)$ 在 K 中的乘法反元素。此时先固定 $\mathbb{F}_{2^n}$ 的一个模型，即

$$\mathbb{F}_{2^n} \cong \mathbb{F}_2[t]/(l(t))$$

其中 $\mathbb{F}_{2^n}[t]$ 表示系数为 $\mathbb{F}_2 = \{0,1\}$ 的多项式集合，即

$$\mathbb{F}_{2^n}[t] = \{f(t) \mid f(t) = \sum_{i=0}^{m} a_i t^i \quad m \text{ 为某非负整数}\}\text{；}$$

而 $l(x)$ 表示一个固定次数（Degree）为 n 的不可约 $\mathbb{F}_2$ 多项式。求乘法反元素就要使用 $\mathbb{F}_2$ 多项式版的广义辗转相除法。考虑如下实例。

例 10.8 令 $K = \mathbb{F}_8$，固定模型

$$\mathbb{F}_8 \cong \mathbb{F}_2[t]/(t^3 + t + 1)\text{，}$$

求 $5 = (101)_2 = t^2 + 1$ 在 K 的乘法反元素。以 $\mathbb{F}_2$ 多项式版广义辗转相除法计算

k	0	1	2	3	
r_k	t^2+1	t^3+t+1	t^2+1	1	0
q_k		0	t	t^2+1	
x_k	1	0	1	t	0

故乘法反元素为

$$t = (10)_2 = 2\text{。}$$

由上例可知，以 $\mathbb{F}_{2^n}$ 为系数的椭圆曲线，可适当地转换成二元码，故在硬件上的设计非常便利。在实际中，定义在 $\mathbb{F}_{2^{160}}$ 之上的椭圆曲线为密码系统常用的曲线。

例 10.9 考虑 Galois 域 $\mathbb{F}_4 \cong \mathbb{F}_2[t]/(t^2 + t + 1)$，任何在 $\mathbb{F}_4$ 的元素均可表示为 2 位的 (0,1) 字串，即

$$\begin{aligned}
0 &= (00)_2 = 0 \\
1 &= (01)_2 = 1 \\
2 &= (10)_2 = t \\
3 &= (11)_2 = t + 1 \\
2^{-1} &= t^{-1} \pmod{t^2 + t + 1} = t + 1 = (11)_2 = 3 \\
3^{-1} &= (t + 1)^{-1} \pmod{t^2 + t + 1} = t = (10)_2 = 2\text{。}
\end{aligned}$$

为方便起见，往往以数字 $\{0,1,2,3\}$ 代表 $\mathbb{F}_4$，而其中的“加法”即 $\mathbb{F}_2$ 多项式的加法，即其二进制代码的 XOR 运算，“乘法”即 $\mathbb{F}_2$ 多项式的乘法再经模运算 $(\bmod\ t^2+t+1)$，结果整理如下：

⊕	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

⊕	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

例 10.10（续上例）　令定义在 $\mathbb{F}_4 \cong \mathbb{F}_2[t]/(t^2+t+1)=\{0,1,2,3\}$ 的椭圆曲线

$$E: y^2+xy=x^3+2\text{。}$$

考虑在 E 上的所有 $\mathbb{F}_4$ 点：

$$x=0 \Rightarrow y^2=2 \Rightarrow y=3$$
$$x=1 \Rightarrow y^2+y=3 \Rightarrow \text{无解}$$
$$x=2 \Rightarrow y^2+2y=0 \Rightarrow y=1,3$$
$$x=3 \Rightarrow y^2+3y=3 \Rightarrow \text{无解}$$

故

$$E(\mathbb{F}_4)=\{(0,3),(2,1),(2,3),\mathcal{O}\}$$

共有 4 点。

考虑 $E(\mathbb{F}_4)$ 的群结构，先计算 $R=(0,3)+(2,3)$

$$m = (3\oplus 3)(0\oplus 2)^{-1}=0\ ,$$
$$x(R) = 0\oplus 2\oplus 0=2\ ,$$
$$y(R) = 2\oplus 3=1\ ,$$

故得 $(0,3)+(2,3)=(2,1)$，另外

$$(2,3)+(2,1)=\mathcal{O}\,,[2](0,3)=\mathcal{O}\,,[2](2,3)=(0,3)\,,[2](2,1)=(0,3)\,,$$

可整理如下：

+	$\mathcal{O}$	(0,3)	(2,1)	(2,3)
$\mathcal{O}$	$\mathcal{O}$	(0,3)	(2,1)	(2,3)
(0,3)	(0,3)	$\mathcal{O}$	(2,3)	(2,1)
(2,1)	(2,1)	(2,3)	(0,3)	$\mathcal{O}$
(0,3)	(2,3)	(2,1)	$\mathcal{O}$	(0,3)

其中各元素的秩分别为

$$\text{ord}(\mathcal{O})=1\,,\ \text{ord}((0,3))=2\,,\ \text{ord}((2,3))=4\,,\ \text{ord}((2,1))=4\,,$$

故点 $(2,3)$ 与点 $(2,1)$ 为生成元，而 $E(\mathbb{F}_4)$ 与 $\mathbb{Z}/4$ 同构，即

$$f\text{、}g: \mathbb{Z}/4 \to E(\mathbb{F}_4) \quad \text{其中 } f(n)=[n](2,3)\text{、}\ g(n)=[n](2,1)$$

均为群同构。

另外 $E(\mathbb{F}_{2^n})$ 的点个数估计，也可由完整版的 Hasse 定理给出答案。

定理 10.6（Hasse） 令 K 为个数为 $q=p^n$（p 为质数）的 Galois 域，令 E 为定义在 K 上的椭圆曲线，则

$$\#E(K)=q+1+t$$

其中误差项

$$|t|\leqslant 2\sqrt{q}\text{。}$$

10.5 密码安全曲线

密码系统所用的椭圆曲线，基本是考虑该曲线上的离散对数问题，以质数曲线为例。

$$E(p,a,b):y^2\equiv x^3+ax+b \pmod p,$$

其中 p 为质数，而 a、b 为整数，且 P 为 $E(p;a,b)$（$\mathbb{F}_p$）上的一点，其秩（Order）（即最小正整数 n 使得 $[n]P=\mathcal{O}$）为 ord(P)。依据 Lagrange 定理

$$\text{ord}(P)\,|\,g=\#E(p;a,b)(\mathbb{F}_p),$$

原则上点 P 的选择要使得值

$$h=\frac{g}{\text{ord}(P)}$$

越小越好。注[11]若 $E(p;a,b)$ 的另一点 Q 为 P 所生成，离散对数问题就是要解 $[n]P=Q$ 中的 n。椭圆曲线的离散对数问题一般要比 $\mathbb{Z}/p^x$ 的离散对数问题困难，后者有所谓的 Calculus Index 法，而一般椭圆曲线迄今为止却没有类似成功的算法，但是当椭圆曲线 $E(p;a,b)$ 是所谓的 Anomalous，即

$$\#E(p;a,b)(\mathbb{F}_p)=p$$

或是所谓的 Supersingular，即

$$\#E(p;a,b)(\mathbb{F}_p)=p+1$$

时，都有较有效的算法攻击，故此两类曲线在密码系统使用上都应排除。另外，$g=\#E(p;a,b)(\mathbb{F}_p)$也应有大的质因数因子。否则，若只有小质因子，只需要使用中国余式定理、Pohlig-Hellman 算法攻击，也是不安全的。由于椭圆曲线密码系统以密钥 163 位就可达到 RSA 密钥 1024 位的安全强度，因此，在把握使用“安全曲线”的原则上，是否一定要与其他安全性强度较弱的密码原件，如 128 位的 Hash 函数，或是配合短密钥对称性密码系统共同使用，成为一个令人深思的问题。

例 10.11 令 $p>3$ 为质数、b 为整数，且 $p\,|\,b$。令椭圆曲线

$$E:y^2\equiv x^3+b \pmod p \text{ 其中 } p\equiv 2 \pmod 3,$$

注[11] 当然最好是 $g=\text{ord}(P)$，但 $E(p;a,b)(F_p)$ 并不一定是循环群，此时就无生成元（类似 Z/p^x 的原根），但至少可以确定

$$E(p;a,b)(F_p)\cong Z/n_1\times Z/n_2,$$

其中 $n_1|n_2$，（若 $n_1=1$，则 $E(p;a,b)(F_p)$ 为循环群）要克服此问题就必须深入椭圆曲线理论，可参阅 Joseph H.Silverman 的“The Arithmetic of Elliptic Curves”Springer Verlag，1985 一书。

则 E 为 Supersingular 曲线。

证明：只需证明 $\#E(\mathbb{F}_p)=p+1$；此处我们运用群论证明，对群论不熟的读者可跳过这段说明。

考虑乘法群的 Homomorphism：

$$\begin{array}{ccccc} \ker & \xrightarrow{\text{id}} & \mathbb{F}_p^\times & \xrightarrow{\varphi} & \mathbb{F}_p^\times \\ x & \mapsto & x & \mapsto & x^3 \end{array}$$

其中 $\ker:=\{x\in\mathbb{F}_p^\times \mid x^3\equiv 1 \pmod p\}$（$\#\ker=1$ 或 3）为乘法群 $\mathbb{F}_p^\times$ 的子群，故由 Lagrange 定理，可得

$$\#\ker \mid \#\mathbb{F}_p^\times = p-1\equiv 1 \pmod 3 \quad (\because p\equiv 2 \pmod 3),$$

故 $\ker=\{1\}$，因此 $\varphi(x)=x^3$ 为乘法群 $\mathbb{F}_p^\times$ 上自身的群同构，换句话说，每个 $y\in\mathbb{F}_p$ 都有惟一存在的三次方根 $x'\in\mathbb{F}_p$，因此对每个 $y\in\mathbb{F}_p$，都有 $y'=y^2\in\mathbb{F}_p$，对 $y''=y'-b$ 都有惟一存在的三次方根 $x\in\mathbb{F}_p$；简言之，每个固定 $y\in\mathbb{F}_p$，都有惟一存在的点 $(x,y)\in E(\mathbb{F}_p)$，再加上无穷远点 $\mathcal{O}$，共 $p+1$ 点。

10.6　将信息转化为椭圆曲线代码

前面所提的公开密钥密码系统，都是构建在 $\mathbb{Z}/n$ 之上，将其信息转化为数字代码。较为经济的方式，不外乎是选取适当的基底 B，将信息 m 转化为 B 进制，即

$$m=m_0+m_1B+m_2B^2+\cdots+m_NB^N \qquad （当中 B^{N+1}\leqslant n）。$$

然而在椭圆曲线上，将信息编码成为适当的点代码，并由点代码转化为信息，就不是如此地容易。以笔者所知，没有能在多项式时间内的确定性算法能够如此互相转换编码，一般而言，采取 Kobliz 建议的一种概率式算法编码。所谓概率式，就是会有部分信息无法编码，实际中的考虑无法编码信息的概率为 2^{-30}。考虑椭圆曲线

$$E: y^2\equiv x^3+ax+b \pmod p,$$

而信息 m 将对应到 $E(\mathbb{F}_p)$ 的某点的 x 坐标值。然而，m^3+am+b 为完全平方数的概率为 $\frac{1}{2}$，因此可添加若干位在 m 上，即

$$m'=mK+j \quad (j=0,\cdots,K-1)$$

使得 x^3+ax+b 为完全平方数，如此便可对应到 $E(\mathbb{F}_p)$ 上的一点；而失败的概率，即 m' 无法对应到 $E(\mathbb{F}_p)$ 的概率是为 2^{-K}，其中 $(m+1)K<p$。

而接收方收到信息 $P_m=(x,y)$ 只需计算

$$m=[\frac{x}{K}]$$

就可回复信息。注意此处只是编码，并未涉及加密解密。

例 10.12　以椭圆曲线

$$E: y^2\equiv f(x)=x^3+82x+502773 \pmod{502807}$$

为例，欲将明文 m = 'coding' 用曲线上的点表达，其中表达失败概率为 $\frac{1}{2^{20}}$，取 K = 20。约定代码为：空白 = 0、a = 1、b = 2、…、z = 26，
因

$$[\log_{27}\frac{502807}{20}] = 3,$$

故每 3 个字母区块编码

$$\begin{aligned} m &= '\text{coding}' \\ &= (m_1, m_2) \\ &= ((3, 15, 4)_{27}, (9, 14, 7)_{27}) \\ &= (2596, 1047), \end{aligned}$$

其中

$$x(P_{m_1}) = f(20m_1),\ x(P_{m_2}) = f(20m_2 + 3)$$

在 $\mathbb{F}_p^{\times}$ 有平方根，即 Legendre 符号

$$\left(\frac{51920}{p}\right) = \left(\frac{2043}{p}\right) = 1。$$

另外由于 $p \equiv 3 \pmod 4$，计算

$$y(P_{m_1}) = 51920^{\frac{p+1}{4}} \pmod p = 309270,$$

$$y(P_{m_2}) = 20943^{\frac{p+1}{4}} \pmod p = 142027,$$

得曲线点代码

$$P_{m_1} = (51920, 309270),$$

$$P_{m_2} = (20943, 142027),$$

而曲线点代码化为数字代码只需要计算

$$m_1 = [\frac{x(P_{m_1})}{20}] = [\frac{51920}{20}] = 2596,$$

$$m_2 = [\frac{x(P_{m_2})}{20}] = [\frac{20943}{20}] = 1047。$$

10.7 椭圆曲线公开密钥密码算法

由于椭圆曲线离散对数问题，只要很短的密钥，就可达到 RSA 一般长度密钥的安全性，所以有些密码专家将其比喻为密码学的明日之星。综观所有以乘法群 $\mathbb{Z}/p^{\times}$ 离散对数为基础的公开密钥密码系统，均可改为相对的椭圆密码系统；虽然也曾考虑 RSA 在椭圆曲线的推广，但后来有人指出其实与 RSA 的计算安全度同数量级，所以不考虑实际应用问题，只有纯学术价值。以 Diffie-Hellman 密钥交换而言，只需将 $\mathbb{Z}/p^{\times}$ 换成椭圆曲线 $E/\mathbb{F}_q$（当中 q = p 质数或

$q=2^m$），将数 g 的 A 次幂方运算 $g \mapsto g^A \pmod p$ 换成点 P 的 A 倍运算 $P \mapsto [A]P$ 即可。以椭圆曲线离散对数问题为基础的公开密钥密码系统，都要先约定曲线参数值：

- 先约定要计算的椭圆曲线 $E=E(q;a,b)$，
  若 $q=p$ 为质数，即 $E: y^2 = x^3+ax+b \pmod p$；
  若 $q=2^n$ 时，即 $E: y^2+xy = x^3+ax^b+b/\mathbb{F}_{2^m}$，选取 Galois 域的模型 $\mathbb{F}_{2^n} \cong \mathbb{F}_2[t]/(l(t))$（其中 $l(t)$ 为次数 n 的不可约的 $\mathbb{F}_2$ 多项式）
- 计算值 $g=\#E(\mathbb{F}_q)$。
- 选择在 $\mathbb{F}_q$ 上的某点 P，使得
  值
  $$n=\mathrm{ord}(P)$$
  有大质数因子。
  值
  $$h=\frac{\#E(\mathbb{F}_q)}{\mathrm{ord}(P)}$$
  很小。
- 曲线参数值 $(E/\mathbb{F}_q,P)$ 代表 $(q,l(x),a,b,g,x(P),y(P),\mathrm{ord}(P),h)$。

例 10.13（质数曲线简例）　选取一条 20 位质数曲线

$$E: y^2 \equiv x^3+ax+b \pmod p$$

含点 $P=(1,65537)$。

（1）随机取一个 20 位质数

$$p=681899\text{。}$$

（2）随机取整数

$$a=251114\text{，}\quad p=65537^2-1-a=247352\text{，}$$

且

$$\gcd(4a^3+27b^2,p)=1\text{。}$$

（3）因为 p 小，可利用 Legendre 符号公式计算

$$g=\#E(\mathbb{F}_p)=p+1+\sum_{x=0}^{p-1}\left(\frac{x^3+ax+b}{p}\right)=681657\text{。}$$

（4）质因数分解 g，得

$$681657=3\times 227219\text{。}$$

（5）计算

$$[227219]P=[227219](1,65537)=\mathcal{O}\text{，}$$

故点 $P=(1,65537)$ 的秩为

$$n=\mathrm{ord}(P)=227219\text{，}$$

$$h=\frac{\#E(\mathbb{F}_p)}{\mathrm{ord}(P)}=3\text{。}$$

算法 10.1（椭圆曲线版的 Diffie-Hellman 密钥交换）　Alice 与 Bob 要用椭圆曲线密码共

同约定密钥。

- （曲线参数约定）约定 $(E/\mathbb{F}_q, P)$。
- Alice 选择一正整数 A，计算 $P_A=[A]P$，将点 P_A 传送给 Bob。
- Bob 选择一正整数 B，计算 $P_B=[B]P$，将点 P_B 传送给 Alice。
- Alice 收到 P_B，计算 $P_K=[A]P_B$，P_K 即共同约定的密钥。
- Bob 收到 P_A，计算 $P_K=[B]P_A$，P_K 即共同约定的密钥。

例 10.14（EC Diffie-Hellman 密钥交换简例） Alice 与 Bob 要用椭圆曲线密码共同约定密钥。

- 曲线参数协定如例 10.12

$$E: y^2 \equiv x^3+251114x+247352 \pmod{681899},$$
$$P=(1,65537),$$
$$g=\#E(\mathbb{F}_p)=681657,$$
$$n=\mathrm{ord}(P)=227219,$$
$$h=\frac{\#E(\mathbb{F}_p)}{\mathrm{ord}(P)}=3\text{。}$$

- Alice 选择一正整数 $A=99138$，计算

$$P_A=[A]P=[99138](1,65537)=(410541, 668940)$$

  将 P_A 传送给 Bob。
- Bob 选择一正整数 $B=12570$，计算

$$P_B=[B]P=[12570](1,65537)=(184674, 237849)$$

  将 P_B 传送给 Alice。
- Alice 收到 P_B，计算

$$P_K=[A]P_B=[99138](184674, 237849)=(27978, 190758),$$

  P_K 即共同约定的密钥。
- Bob 收到 P_A，计算

$$P_K=[B]P_A=[12570](410541, 668940)=(27978, 190758),$$

  P_K 即共同约定的密钥。

椭圆曲线的 ElGamal 密码只是将原先在 $\mathbb{Z}/p^\times$ 的指数运算换成在椭圆曲线 $E/\mathbb{F}_q$ 上的加法运算而已。

算法 10.2（椭圆曲线版的 ElGamal 密码） Alice 欲明文代码 P_m 加密成密文 P_c 传送给 Bob。

- （曲线参数约定）约定 $(E/\mathbb{F}_q, P)$。
- （密钥产生）Bob 随机选取一个整数 B 且 B 与 g 互质，即 $\gcd(B,g)=1$，计算 $P_B=[B]P$（同椭圆曲线版的 Diffie-Hellman 算法）。公开密钥为 $(E/\mathbb{F}_q, P, P_B)$，私钥为 B。
- （加密）明文代码 $P_m \in E(\mathbb{F}_q)$。Alice 取得 Bob 的公开密钥 $(E/\mathbb{F}_q, P, P_B)$，随机选取整数 A，计算 $P_A=[A]P$（同椭圆曲线版的 Diffie-Hellman 算法），计算 $P_c=P_m+[A]P_B$，密文为 (P_A, P_C)，传送给 Bob。
- （解密）Bob 收到密文 (P_A, P_C)，计算 $P_m=P_c-[B]P_A$ 得明文代码。

证明：$P_m = P_c - [B]P_A$：

因为

$$
\begin{aligned}
P_c - [B]P_A &= (P_m + [A]P_B) - [AB]P \\
&= P_m + [AB]P - [AB]P \\
&= P_m + \mathcal{O} = P_m
\end{aligned}
$$

例 10.15（EC ElGamal 密码简例） Alice 要将明文代码 P_m 加密成密文 P_c 传送给 Bob。

- （曲线参数约定）约定曲线

$$E: y^2 \equiv x^3 + 3x + 498917 \pmod{744187}$$

其中

$$
\begin{aligned}
&P = (283426, 64607) \\
&g = \#E(\mathbb{F}_{744187}) = 744994 \\
&n = \mathrm{ord}(P) = 372497 \text{ 为质数} \\
&h = g/n = 2
\end{aligned}
$$

- （密钥产生）Bob 随机选取一整数 $B = 366161$ 且 $\gcd(B,g) = 1$，计算

$$P_B = [B]P = (9358, 353904),$$

公开密钥为 $(E/\mathbb{F}_q, P, P_B)$，私钥为 B。

- （加密）明文代码

$$P_m = (678189, 436766) \in E(\mathbb{F}_q)。$$

Alice 得 Bob 的公开密钥，随机选取整数 $A = 139523$，计算

$$P_A = [A]P = (568668, 660971),$$

计算

$$
\begin{aligned}
P_C = P_m + [A]P_B &= (678189, 436766) + [139523](9358, 353904) \\
&= (678189, 436766) + (213845, 28661) \\
&= (742504, 399355),
\end{aligned}
$$

密文为 (P_A, P_C)，传送给 Bob。

- （解密）Bob 收到密文 (P_A, P_C)，计算

$$
\begin{aligned}
P_m &= P_C - [B]P_A \\
&= (742504, 399355) - [366161](568668, 660971) \\
&= (742504, 399355) - (213845, 28661) \\
&= (742504, 399355) + (213845, -28661) \\
&= (678189, 436766)
\end{aligned}
$$

得明文代码。

算法 10.3（椭圆曲线的 ElGamal 数字签名） Alice 要将信息 m 数字签名成 s 传送给 Bob，其中 m 为整数且 $0 \leqslant m \leqslant n$。

- （曲线参数约定）约定 $(E/\mathbb{F}_q, P)$。
- （密钥产生）Alice 随机选取一个整数 A，计算 $P_A = [A]P$（同椭圆曲线版的 Diffie-Hellman 算法）。公开密钥为 $(E/\mathbb{F}_q, P, P_A)$，私钥为 A。
- 数字签名

（1）Alice 随机选取一整数 k 使得 $\gcd(k, g) = 1$。

（2）计算 $R=[k]P$ 。

（3）计算 $s^*=k^{-1}(m-Ax(R)) \pmod n$，其中 $x(R)$ 为点 R 的 x 坐标。

（4）将数字签名

$$s=(m,R,s^*)$$

传送给 Bob。

- 验证

（1）Bob 收到数字签名 $s=(m,R,s^*)$，并取得 Alice 的公钥 $(E/\mathbb{F}_q,P,P_A)$ 。

（2）计算

$$V_1=[x(R)]P_A+[s^*]R,$$
$$V_2=[m]P。$$

（3）若 $V_1=V_2$ 则验收，否则拒绝。

证明：$V_1=V_2$：

$$\begin{aligned}V_1&=[x(R)]P_A+[s^*]R\\&=[x(R)][A]P+[k^{-1}(m-Ax(R))][k]P\\&=[x(R)A+m-Ax(R)]P\end{aligned}$$

$$(\because [k^{-1}k]P=[1+gt]P=P+[t]([g]P)=P+\mathcal{O}=P \text{ 其中 } t\in\mathbb{Z})$$

$$=[m]P=V_2。$$

ElGamal 加密与 ElGamal 数字签名的对象不同；ElGamal 加密的对象是明文代码点

$$P_m\in E(\mathbb{F}_q),$$

而 ElGamal 数字签名的对象是信息代码

$$m\in\mathbb{Z}/n。$$

对于以椭圆曲线对数问题为主的附录式数字签名方案均是如此。

例 10.16（EC ElGamal 数字签名简例） Alice 欲数字签名信息 $m=204819$ 成 s 传送给 Bob，其中 m 为整数且 $0\leqslant m\leqslant n$ 。

- （曲线参数约定）约定曲线

$$E: y^2\equiv x^3+13x+445618 \pmod{581227}$$
$$P=(1,99787)$$
$$g=\#E(\mathbb{F}_p)=n=\mathrm{ord}(P)=580373$$
$$h=1。$$

- （密钥产生）Alice 随机选取一个整数 $A=247070$，计算

$$P_A=[A]P=(535932, 400788),$$

公开密钥为 $(E/\mathbb{F}_q,P,P_A)$，私钥为 A。

- 数字签名

（1）Alice 随机选取一个整数 $k=202277$ 使得 $\gcd(k,n)=1$ 。

（2）计算

$$R=[k]P=(107521, 116109)。$$

（3）计算

$$\begin{aligned} s^* &= k^{-1}(m - Ax(R)) \pmod n \\ &= 212315(204819 - 247070 \times 107521) \pmod{580373} \\ &= 210277。\end{aligned}$$

（4）将数字签名

$$s = (m, R, s^*)$$

传送给 Bob。

- 验证

（1）Bob 收到数字签名并取得 Alice 的公钥。

（2）计算

$$V_1 = [x(R)]P_A + [s^*]R = (291788, 530336)$$
$$V_2 = [m]P = (291788, 530336)。$$

验收。

算法 10.4（椭圆曲线版的 DSA，ECDSA） Alice 要将信息 m 数字签名成 s 传送给 Bob，其中 m 为整数且 $0 \leqslant m \leqslant n$。

- （曲线参数约定）约定 $(E/\mathbb{F}_q, P)$。
- 密钥产生

（1）Alice 找出 $g = \#E/\mathbb{F}_q$ 的大质数因子（假设 $0 \leqslant m \leqslant n$）。

（2）找出点 $P_A \neq \mathcal{O} \in E/\mathbb{F}_q$ 使得 $[n]P_A = \mathcal{O}$。

（3）Alice 选择一整数 A，计算 $P_B = [A]P_A$。

（4）Alice 公布公钥 $(E/\mathbb{F}_q, P, n, P_A, P_B)$，私钥为值 A。

- 数字签名

（1）Alice 随机选取一个整数 k 使得 $1 \leqslant k \leqslant n$。

（2）计算 $R = [k]P_A = (x(R), y(R))$。

（3）计算 $s^* = k^{-1}(h(m) + Ax(R)) \pmod n$，其中 $x(R)$ 为点 R 的 x 坐标，$h(\cdot)$ 表示 Hash 函数。

（4）将数字签名

$$s = (m, R, s^*)$$

传送给 Bob。

- 验证

（1）Bob 收到数字签名 $s = (m, R, s^*)$，并取得 Alice 的公钥 $(E/\mathbb{F}_q, P, n, P_A, P_B)$。

（2）计算

$$v_1 = s^{*-1} h(m) \pmod n$$
$$v_2 = s^{*-1} x(R) \pmod n$$
$$P_V = [v_1]P_A + [v_2]P_B$$

（3）若 $P_V = R$ 则验收，否则拒绝。

证明：$[s^*]P_V = [s^*]R$：

$$
\begin{aligned}
s * P_V &= [s^*]([v_1]P_A + [v_2]P_B) \\
&= [h(m)]P_A + [x(R)]P_B \\
&= [h(m)]P_A + [x(R)]([A]P_A) \\
&= [h(m) + Ax(R)]P_A \\
&= [s * k]P_A \\
&= [s^*]R\ 。
\end{aligned}
$$

注（概率上的错误） 与 DSA 一样，在密钥产生时，s^* 有可能会使得 $s^* \equiv 0 \pmod n$，此时当然不存在 $s^{*-1} \pmod n$，因此必须重新选取 k，但出现如此情形的概率为 $\frac{1}{n}$，是微乎其微的。一般而言，当签署者发现 $s^* = 0$ 或 $R = \mathcal{O}$ 时就要重新选取 k。

10.8 椭圆曲线因数分解

利用椭圆曲线的代数群结构，其实也可用来作因数分解，这可追溯到 Lenstra 的一篇论文，注[12]其基本想法就是将 Pollard 的 $p-1$ 法推广至椭圆曲线。以 $n = pq$ 为例（p、q 均为质数），由中国余式定理，可将乘法群 $\mathbb{Z}/n^\times$ 分解成

$$\mathbb{Z}/n^\times \cong \mathbb{F}_p^\times \times \mathbb{F}_q^\times\ 。$$

Pollard 的 p－1 法就是考虑 $a \in \mathbb{Z}/n^\times$，它的某个 $p-1$ 的倍数幂次会同余 1，即

$$a^{m(p-1)} \equiv 1 \pmod n,$$

其中 m 为某整数，并预期

$$\gcd(a^{m(p-1)} - 1, n) = p,$$

而 $p-1$ 若是“Smooth”，注[13]Pollard 的 $p-1$ 法就有可能成功。

而椭圆曲线的情形，就是用椭圆曲线取代 Pollard 的 $p-1$ 法的乘法群，同样借助中国余式定理，可得

$$E(\mathbb{Z}/n) \cong E(\mathbb{F}_p) \times E(\mathbb{F}_q),$$

其中 E 为某系数为整数的椭圆曲线（为方便模运算），此时选取某点 $P = [x, y, 1]$（为加权投影坐标值）。假设

$$\# E(\mathbb{F}_p)$$

为 Smooth，适当地选取整数值 k，计算

$$[x_k, y_k, z_k] \equiv [k!]P \pmod n\ 。$$

若

$$\# E(\mathbb{F}_p) \mid k!$$

则 p 整除 z_k，此时，n 的因子可通过计算

$$\gcd(z_k, n)$$

注[12] 参阅 H.W.Lenstra 的“Factoring integers with elliptic curves” Annals of Math.126(1987)，P.649-673。

注[13] p-1 是“Smooht”当“p-1”只有“小”质因子。

求出。值得注意的是，当 $\gcd(z_k,n)>1$ 时，加权投影坐标值 $[x_k,y_k,z_k] \pmod n$ 是不代表 $E(\mathbb{F}_p)$ 中任何一点的。若以平面坐标计算，就必须额外处理，这也是以加权投影坐标计算的好处之一。另外由 Hasse 定理知

$$p-2\sqrt{p} \leqslant \#E(\mathbb{F}_p) \leqslant p+2\sqrt{p},$$

$\#E(\mathbb{F}_p)$ 的值分布在"小"范围中，而且会随 a、b 变动而"均匀"分布，注[14]这使得椭圆曲线因数分解法又多具备了概率的成分。以上为椭圆曲线因数分解法的原理，在某些数学软件如 Maple、Mathematica 中都已有现成的命令，对于 20～50 的十进位的数的因数分解有实质的作用，由于曲线的选取可任意随机选取，且相互无关联性，故可以并行计算处理，这在实际计算中有相当价值，但前提是数值 $\#E(\mathbb{F}_p)$ "Smooth"，点 P 的选取要适当，这使得椭圆曲线因数分解不一定能"成功"，然而有时对很大的数反而有效，如 617 的十进位数费马 11 号数

$$\mathbb{F}_{11}=2^{2^{11}}-1=2^{2048}-1$$

就被 Bent 于 1989 年用椭圆曲线因数分解法分解成功，注[15]椭圆曲线因数分解法具有概率算法的特性。一般而言，其算法在仅使用一条曲线的计算复杂度为

$$O(e^{\sqrt{2\ln p \ln\ln p}}),$$

（其中 p 为 n 的最小质因子）若 $p \approx \sqrt{n}$ 时，这比二次筛法缓慢，但仍属同数量级。

算法 10.5（椭圆曲线因数分解）　令 $n \geqslant 2$ 为一个合数，欲找出 n 的某因数。

```
ECFactor(n)
{
    if(g=gcd(n,6)>1)
        return g;
    m=n;
    for(r=2;m>=2;r++)
    {
        m=n^(1/r);
        if(isInteger(m))
            return m;
    };
//判断 n 是否与 6 互质或 n 非某整数 m 的 r 次幂
Point:
    px=RandomInteger(1,n);
    py=RandomInteger(1,n);
//P=(px,py)=[px,py,1]为椭圆曲线的点
newCurve:
    a=RandomInteger(1,n);
    b=py^2-px^3-a*px(mod n);
//E:y^2=x^3+a*x+b
Discriminant:
```

注[14] 参阅 J.F.Blake、G.Seroussi 与 N.P.Smart 的 "Elliptic Curves in Cryptography" Cambridge University Press，（1999）P.161。

注[15] 参阅 D.M.Bressoud 的"Factorization and Primality Testing,"Springer-Verlag，New York，（1989）。

```
    g=gcd(4*a^3+27*b^2,n);
    if(g==n)
        goto newCurve;
    else if(1<g<n)
        return g;
        //g=1
new:
    input>>k;
    //选取适当的 k 值
compute:
    [x,y,z]=[px,py,1];
    factor=1;
    for(B=2;B<=k;B++)
    {
        [x,y,z]=[B][x,y,z] (mod n);
        //以加权投影坐标公式计算
        factor=gcd(n,z);
        if(1<factor<n)
            return factor;
        else if(factor==0)
            goto newCurve;
        else
        //factor==1
            continue;
    };
    if(factor==1)
        goto new;
}
```

例 10.17 令 $n = 24287$，欲以椭圆曲线法因数分解，选取椭圆曲线

$$E: y^2 \equiv x^3 + ax + b \text{ 含点 } P = (13,1),$$

随机取 $a = 13$， $b = 13^2 - a - 1 = 155$。以加权投影坐标计算 $[k!]P \pmod n$（$k = 2,3,4,\cdots$）得

$$[2!]P \pmod n = [2][1, 13, 1] = [51501, 10252, 26]$$
$$[3!]P \pmod n = [3][51501, 10252, 26] = [3653, 31955, 17322]$$
$$[4!]P \pmod n = [4][3653, 31955, 17322] = [4044, 35059, 11405]$$
$$[5!]P \pmod n = [5][4044, 35059, 11405] = [38656, 12056, 267]$$
$$[6!]P \pmod n = [6][38656, 12056, 267] = [45019, 593, 5576]$$
$$[7!]P \pmod n = [7][45019, 593, 5576] = [30419, 46401, 19165]$$
$$[8!]P \pmod n = [8][30419, 46401, 19165] = [44679, 33806, 51197]$$
$$[9!]P \pmod n = [9][44679, 33806, 51197] = [2693, 29599, 6923]$$
$$[10!]P \pmod n = [10][2693, 29599, 6923] = [26157, 12465, 27565]$$

当中逐次计算 $\gcd(n, z([k!]P))$（$k = 2,3,4,\cdots$），在 $k = 10$ 时，得

$$\gcd(n, z([10!]P)) = \gcd(27565, 52597) = 149$$

为 n 的因数，而 $n/149 = 353$ 均为质数，故 n 的质因数分解为

$$52597 = 149 \times 353\text{。}$$

例 10.18　以椭圆曲线因数分解法分解一个 30 位整数 $n = 416824937$ ，我们以 10 条椭圆曲线含点 $P = (1,5)$ 处理，其中

$$E/\mathbb{Z}: y^2 = x^3 + ax + b,$$
$$P_k = [k!]P = [x_k, y_k, z_k],\quad (\gcd(z_k, n) > 1)$$
$$b = 5^2 - a - 1,$$

其计算结果如表 10-2 所示。

表 10-2　例 10.8.3 计算结果

i	a	b	k	P_k	$\gcd(z_k,n)$
1	1	23	43	[274685671, 40571092, 163562272]	21937
2	2	22	46	[353933000, 361170342, 138049541]	21937
3	3	21	79	[124713088, 51546826, 112751934]	19001
4	4	20	353	[196000083, 361673235, 212964396]	21937
5	5	19	199	[3866157,101298769, 310715668]	21937
6	6	18	7	[41057255, 32340878, 366434285]	19001
7	7	17	67	[377489714, 36225653, 368739033]	21937
8	8	16	59	[406999829, 330386496, 20028481]	21937
9	9	15	2087	[292245752, 102729625, 221817674]	21937
10	14	10	11047	[343369149, 160715264, 317932941]	21937

由计算结果可知，椭圆曲线因数分解所需计算的差异度很大，最少只需要 $k = 7$ ，而最多竟达 $k = 11047$ （甚至比试除法还慢）。这就需要利用椭圆曲线法因数分解时多试几条不同的曲线。

10.9　ECCp-109 挑战

加拿大专门研发制造密码产品的公司 Certicom，自 1997 年 11 月 6 日在网站上公布了一系列有关椭圆曲线离散对数问题的挑战，当中所考虑的曲线有质数曲线与二元曲线。各种不同曲线的挑战分别命名为：

- ECCp-x 代表随机的质数曲线，x 代表 x 位的质数。
- ECC2-m 代表随机的二元曲线，m 代表秩为 2^m 的 Galois 域。
- ECCk-m 代表一种特殊的二元曲线，称之为 Koblitz 曲线，而 m 代表秩为 2^m 的 Galois 域。

以 ECCp-109 挑战为例，Certicom 所提供的奖金为一万美金。Certicom 更是将挑战的级数分成练习级、第一级以及第二级等三种。对于第二级的挑战，如 ECCp-163，其奖金就高达三万美金，而最高奖额的为 ECCp-359、ECC2-353。ECC2K-358，奖金均为十万美金。究竟 Certicom 举办这些椭圆曲线问题挑战的目的是什么？据他们自己声称，举办这类挑战的目标有下列 6 项：

（1）增进密码界对椭圆曲线离散对数问题的了解以及、对其困难度的认识。

（2）不同密码系统如 ECC、RSA 以及 DSA 已在理论上有了安全程度的比较，而这项挑战就是要去验证这些理论上的安全比较。

（3）对椭圆曲线公开密钥系统的使用者，提供对特定安全需求所需密钥长度的信息。

（4）判断在质数曲线上以及在二元曲线上的椭圆曲线离散对数问题的困难度是否有重大的区别。

（5）判断在二元曲线上，随机曲线或是 Koblitz 曲线的椭圆曲线离散对数问题的困难度是否有重大的差别。

（6）激励计算数论以及演算数论（Algorithmic Number Theory）的研究，特别是椭圆曲线离散对数问题的研究。

虽然 Certicom 的目标很明确，然而截至目前为止，只有练习级与第一级的 9 项挑战被解出。据 Certicom 估计，以 ECCp-109 挑战为例，若由一部 Pentium100 的电脑来计算，则需要 9.0×10^6 天才会破解，而所需的椭圆曲线计算次数达 2.1×10^{16} 次之多。以下即为 ECCp-109 挑战所公布的数据内容：

```
======== ECCp-109 ========
p = 1BD5 79792B38 0B5B521E 6D9FB599
seedE = 1A4D696E 67687561 5175F8DC 47334A5E EDA1C34C
r = 0167 90D37BA1 CC98B1FE 53558DEA
a = 0FD4 C926FD17 8E9805E6 63021744
b = 153D 3CBB508F FE3A7F31 FF4FAFFD
h = 01
n = 1BD5 79792B38 0B049C4D 13A75AE5
seedP = 2409C5C7 E50FF4D6 96E67687 56151755 2DF2B48C
x = 04CC 974EBBCB FDC3636F EB9F11C7
y = 0761 1B0EB122 9C0BFC5F 35521692
seedQ = 2CC6F5DA 1DBA34D6 96E67687 5615175D 2D30BDF2
x = 0233 857E4E8B 5F005512 6E7D7B7C
y = 19C8 C91063EB 4276371D 68B6B4D9
```

- 其中 p 表示曲线所模的质数。
- Seed E 表示用来生成曲线

$$E(p;a,b): y^2 \equiv x^3 + ax + b \pmod p$$

- 的随机数种子。
- Seed P 表示用来生成点 P 的随机数种子，x、y 是点 P 的 x 坐标、y 坐标。
- n 为点 P 的秩，为质数。
- h 表示数

$$\#E(\mathbb{F}_p)/n\ 。$$

- Seed Q 表示用来生成点 Q 的随机数种子，下面 x、y 即是点 Q 的 x 坐标、y 坐标。
- 其中所有数据皆以 16 进制表示，而随机数生成所使用的算法，则是用到 Hash 函数 SHA-1。

有兴趣的读者可参阅“Certicom ECC Challenge”一文，该文可在 Certicom 官方网站 www.certicom.com/download/did-111/cert.ecc.challenge.pdf 取得。然而对于有心挑战 ECCp-109

的读者，恐怕要失望了！因为在 2002 年 10 月 15 日 12:56 分，椭圆曲线对数问题

$$Q=[k]P$$

的问题已被破解。其解为

$$k = 281183840311601949668207954530684,$$

Certicom 也证实了这项答案。据破解这项挑战的团队的发言人 Chris Monico（当时正在 Notre Dame 大学从事博士后研究）表示：

（1）他们总共让一万部电脑（大部分为 PC）计算了 549 天。

（2）破译成员共有 10308 名成员，分属 249 个单位。

（3）总共计算了 36507222×10^9 个椭圆曲线点，特殊点（Distinguished Point）共 68228567 个。

（4）在领得的一万美元奖金中，他们捐出了 8000 美元给自由软件基金会（Free Software Foundation）；而分别找出最关键步骤的 Meunier Gerard 以及 B.J.Smith 则各得 1000 美元。

然而破译 ECCp-109 的总计算量远高于破解 RSA-155 的计算量，这要比 NIST 的估计还多。事实上，比 ECCp-109 容易的 ECC2-108 就比 RSA-155 难 50 倍，练习级的 ECC2-97 仍然比 RSA-155 难两倍；ECCp-109 是人类目前所能破译公开密钥密码问题的最高纪录，下一阶段的 ECC 131 系列挑战，预计计算量是 ECCp-109 的数千倍，网络上已经有人开始组织，另一个较容易的未破解的挑战为 ECC2-109，极有可能不久就破解，这些都是利用网络分散计算，参加的人分散各大洲。而 Certicom 的 Vanstone 也表示，原先预计与 1024 位密钥 RSA 同级的 163 位密钥 ECC 的计算量将是破译 ECCp-109 的总计算量的10^8 倍。

10.10　并行 Pollard Rho 法

截至目前为止，所有挑战 ECC 公开问题成功的方法，大多是用并行 Pollard Rho 法，其实就是将 Pollard Rho 法改进成便于并行计算的一种算法。而考虑的对象就是椭圆曲线离散对数问题

$$Q=[k]P$$

解 k，其中 $P,Q\in E(\mathbb{F}_q)$，令 G 为由 P 所生成的子群（$\#G=n$）

$$G=\langle P\rangle:=\{[m]P \mid m\in\mathbb{Z}\}\subset E(\mathbb{F}_q),$$

将 G 分成三块大小大致相等的子集S_1、S_2、S_3，随机取两个整数

$$a_0、b_0 使得 1\leqslant a_0，b_0\leqslant n-1。$$

起始点为

$$X_0=[a_0]P+[b_0]Q,$$

构造一个序列（Sequence）$\{X_i\}$ 使得

$$X_i=\begin{cases}P+X_{i-1} & 若 X_{i-1}\in S_1\\ [2]X_{i-1} & 若 X_{i-1}\in S_2\\ Q+X_{i-1} & 若 X_{i-1}\in S_3\end{cases}$$

由此序列可定义两序列$\{a_i\}$与$\{b_i\}$满足

$$X_i=[a_i]P+[b_i]Q,$$

其中

$$a_i = \begin{cases} a_{i-1}+1 \pmod n & 若X_{i-1} \in S_1 \\ 2a_{i-1} \pmod n & 若X_{i-1} \in S_2 \\ a_{i-1} & 若X_{i-1} \in S_3 \end{cases}$$

且

$$b_i = \begin{cases} b_{i-1} & 若X_{i-1} \in S_1 \\ 2b_{i-1} \pmod n & 若X_{i-1} \in S_2 \\ b_{i-1}+1 \pmod n & 若X_{i-1} \in S_3 \end{cases}$$

若 $X_i = X_j\ (i \neq j)$，即找到碰撞，则

$$[a_i]P + [b_i]Q = [a_j]P + [b_j]Q,$$

重新整理上式得

$$k \equiv -\frac{a_i - a_j}{b_i - b_j} \pmod n,$$

若 $b_i - b_j \not\equiv 0 \pmod p$ 就得解。这就是 Pollard Rho 法，Rho 的意义就是序列从起始点开始到找到碰撞 $\{X_i\}$ 的路径会像希腊字母

$$\rho$$

例 10.19（Pollard Rho 法简例） 令椭圆曲线

$$E: y^2 \equiv x^3 + x + 1 \pmod{221},$$

其中 $P = (117,36) \in E(\mathbb{F}_{221})$ 且 $g = \#E(\mathbb{F}_{221}) = 223$ 为质数。另一点 $Q = (29,24) \in E(\mathbb{F}_{221})$，要求解

$$[k]P = Q。$$

随机取整数 $a_0 = 212$， $b_0 = 190$，计算

$$X_0 = [212]P + [190]Q = (205,137)。$$

将 $E(\mathbb{F}_{221})$ 分成三块子集：

$$S_1 = \{(x,y) \in E(\mathbb{F}_{221}) \mid 0 \leqslant x < 70\} \cup \{\mathcal{O}\}$$
$$S_2 = \{(x,y) \in E(\mathbb{F}_{221}) \mid 70 \leqslant x < 140\}$$
$$S_3 = \{(x,y) \in E(\mathbb{F}_{221}) \mid 140 \leqslant x < 221\}。$$

代入计算公式如表 10-3 所示。

表 10-3 例 10.19 用并行 Pollard 计算[k]P=Q 的过程

i	X_i	a_i	b_i	i	X_i	a_i	b_i
0	(205, 137)	212	190	15	(154, 208)	142	201
1	(103, 22)	212	191	16	(171, 59)	142	202

PDG

续表

i	X_i	a_i	b_i	i	X_i	a_i	b_i
2	(177, 139)	201	159	17	(133, 200)	142	203
3	(70, 14)	201	160	18	(65, 179)	61	183
4	(171, 152)	179	97	19	(208, 56)	62	183
5	(154, 3)	179	98	20	(186, 133)	62	184
6	(164, 24)	179	99	21	(122, 152)	62	185
7	(18, 187)	179	100	22	(117, 175)	124	147
8	(125, 119)	180	100	23	(204, 110)	25	71
9	(17, 17)	137	220	24	(112, 186)	25	72
10	(0, 1)	138	200	25	(42,131)	50	144
11	(55, 116)	139	200	26	(123, 56)	51	144
12	(13, 14)	140	220	27	(206, 121)	102	65
13	(200, 145)	141	200	28	(42,131)	102	66
14	(53, 133)	141	201	29	(123, 56)	103	66

发现第一次碰撞发生在 $X_{25}=X_{28}$ ，在 $i=25$ 之后，$X_{i+3}=X_i$，形成周期变化现象，其中

$$a_{25}=50, b_{25}=144, a_{28}=102, b_{28}=66,$$

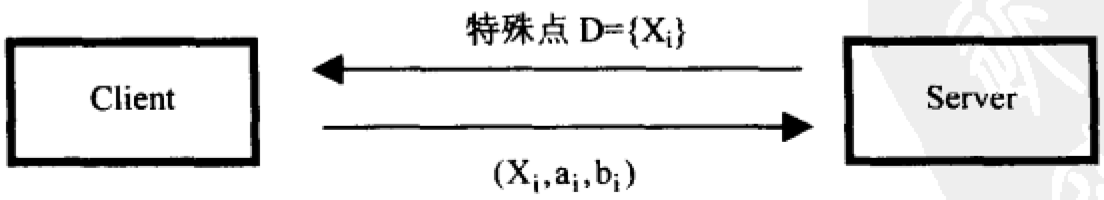
计算

$$k \equiv -(50-102)(144-66)^{-1} \pmod{221} = 75。$$

而并行 Pollard Rho 法是由 Oorschot 以及 Wiener 改良上述方法而成。其做法就是先在群 G 中取一子集 D，其中的元素称为特殊点（Distinguished Point）。每个 Client 计算一个序列$\{X_i\}$直至找到一特殊点$X_i \in D$，将此X_i与所对应的值a_i，b_i传给 Server，而 Client 再重新从新的起始点开始计算。Server 储存所有 Client 所传的点的数据，直至有某点的数据被传送两次，即产生碰撞，再计算最后一个步骤，便可得解。总计算时间约为

$$\left(\frac{\sqrt{\pi n}}{\sqrt{2m}}+\frac{1}{\theta}\right)t$$

其中假设 m 台电脑等速并行计算，θ 为特殊点所占的比例，t 为计算一曲线点的时间。



第 11 章　公开密钥基础建设

在公开密钥技术下，Alice 用她的密钥数字签名一份电子文件 m 成 s 传递给 Bob，Bob 如何取得 Alice 的公钥去解读这份数字签名文件 s？关键是如何能证实所取得的公钥确定是 Alice 的，有这样的不可否认性才有意义。是否需要所谓的“受信任的第三者”（Trusted Third Party，TTP）担任认证机构（Certification Authority，CA）才能证实 Alice 公钥的真实性？当中的信任机制又如何建设？适当的公开密钥基础设施（Public Key Infrastructure，PKI）在此就越发显得重要。

在本章中，除了简介 PKI 外，还要探讨 PKI 协议 X.509，值得注意的是目前几乎所有与 PKI 有关的标准，都是在 X.509 上发展出来的；另外最近最热门的“自然认证”更是不得不提。

11.1　认证机构 CA

如果 Alice 使用某公开密钥密码系统，只是将她的私钥保密是不足够的，若她想与 Bob 通信，她还必须取得 Bob 的公钥，而且还必须知道所取得的公钥确实是 Bob 的。一种解决此问题的方式，就是建立一个双方都信赖的机构，即认证机构 CA，当然所有使用者，均信任 CA。由 CA 的数字签名，CA 可认证所有使用者的公钥的正确性及有效性，所有使用者均知道 CA 的公钥，因此他们均可验证 CA 的数字签名。在 CA 下的公开密钥密码系统的简易运作方式如下：

- 所有使用者均有 CA 公钥资料的有效备份。
- CA 的工作，即用可回复式数字签名方案数字签名下列资料

  （Bob，Bob 的公钥 K_{e_B}），

  此资料连同其数字签名，称之为 Bob 的数字认证（Digital Certificate）。CA 只在已确认公钥确实属于 Alice 的情形下，才会数字签名此资料。
- Alice 欲在此密码系统下加密传递给 Bob，她在 CA 处取得 Bob 的数字认证，用 CA 的公钥验证是否为真。若为真，Alice 就取得 Bob 的真实公钥 K_{e_B} 。

当然实际的数字认证，不会只是

（Bob，Bob 的公钥 K_{e_B} ）

的数字签名。一般而言数字认证可能包括下列信息：

- 使用者姓名。
- 使用者公钥。
- 用来加密或数字签名的密钥。
- CA 的名称。
- 认证的序号。
- 认证到期的时间。

  ⋮

然而数字认证的数据都是公开的，这就同时提供给有心攻击者不少信息，如公钥使用人的姓名、电话号码等，这些信息是否也要放在数字认证中，也有疑虑。

CA 可按功能分成两部分，一是处理使用者身份的认证，二是实际将使用者的公钥数字签名成数字认证；产生数字认证的工作由 CA 完成，而使用者身份的验证，也在其他的机构完成，即注册机构（Registration Authority,RA），大多数 CA 都是采取 CA 与 RA 分开的做法，这样似乎可以在更安全的环境下保护 CA 免受攻击。

在数字认证的有效期内，若其使用者的私钥遭窃、被破解，或其他种种原因，该认证要被注销，而其他使用者也应被告知。一种做法，就是由 CA 定期公布认证中止清单（Certificate Revocation List，CRL），当中包括列出所有被注销数字认证的序号。

例 11.1　Alice 在旅行中，失去了她的 Smart 卡，卡中有 Alice 数字签名的私钥，此时 Alice 本人无法再进行数字签名，因为私钥就仅在这张卡上。如此，Alice 应马上通知 CA，而 CA 必须立即注销 Alice 的数字认证。

曾有人很乌托邦地提出一个构想，即全球成立一个全球 Root CA，每个国家各自设立一个国家 CA，全球 Root CA 将其私钥签发数字认证给各国国家 CA，各国家 CA 再将其私钥签发数字认证给下属 CA，…，最后底层 CA 再将其私钥签发数字认证给该国公民。于是全球 Root CA、各国家 CA 与所有以下多层 CA 以及该国公民也形成树状结构（Tree）。

例 11.2　假设在德国某 CA CA_{DE} 注册的 Alice 想与 Bob 加密通信，先要取得 Bob 的公钥，而 Bob 是在中国台湾地区某 CA CA_{TW} 下注册的用户，Alice 并不知道 CA_{TW} 的公钥。在中国的某 CA CA_{CN} 与 CA CA_{DE}、CA CA_{TW} 皆有对方公钥，Alice 可通过两种不同路径取得 Bob 的公钥，是采用认证链接（Certicate Chain）的方式取得（以 CA≪U≫代表以 U 在 CA 下的数字认证）。

- CA_{DE}≪CA_{CN}≫，CA_{CN}≪CA_{TW}≫，CA_{TW}≪Bob≫。
- CA_{DE}≪CA_{TW}≫，CA_{TW}≪Bob≫。

如果 Alice 不愿相信 CA CA_{CN}，她当然可以选择较短的“信任路径”（Trust Path）。

例 11.3（PGP 信任模式）　eMail 加密软件 PGP 的公开密钥基础设施，就不是以 CA 为主的树状结构。相反他是以个别使用者为中心，个别使用者 Alice，可依自行需求设定对其他使用者 Bob 的不同信任程度：

- （完全信任，Complete Trust）：Alice 完全信任 Bob，所有由 Bob 密钥所签署的密钥，她都信任。
- （部分信任，Partial Trust）：Alice 不完全信任 Bob，所有由 Bob 密钥所签署的密钥，还需要别人的签署，她才可能信任。
- （不信任，No Trust）：Alice 完全不信任 Bob，所有由 Bob 密钥所签署的密钥，她都不信任。
- （未知，Unknown）：Alice 不知要不要信任 Bob，通常是视为 Alice 不信任 Bob。

加密软件 PGP 就是依照这些不同信任程度的模式，在不同使用者之间，形成一个信任网（Trust Web），由此有效地安全加密通信信息。然而这样的信任模式，是不太适合电子商务的。

11.2 X.509

CA 核发数字认证的格式，目前都是以 X.509 的规格为主。X.509 是利用公开密钥密码技术对通信的双方订立了几种不同识别的方法。最初的版本发表于 1988 年，经多年的修改，最新版本发表于 1997 年，目前所使用的版本 V3 版为 ISO9594-8 的认证标准。基本的 X.509 数字认证结构很简单，但在高级应用上，因为要加入额外的信息，可能会很复杂，一般而言，标准文件中所定义的 X.509 数字认证内容，包括下列信息：

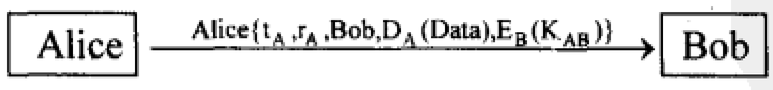
$$CA \ll A \gg = CA\{V,SN,AI,CA,UCA,A,UA,A_p,T^A\}$$

其中各符号所代表的意义如下：

- $CA \ll A \gg$：由认证机构 CA 发给使用者 A 的数字认证。
- V：版本号数。用来区别是哪一版本 X.509 所作的数字认证。
- SN：序号，每个不同的使用者都有惟一的序号。
- AI：算法的识别代码，CA 是用何种数字签名的算法来制作该数字认证的。
- CA：CA 的名称。
- UCA：CA 的特有编号（也可选择不用）。
- A：使用者，即此公钥的拥有者。
- UA：使用者识别码，此公钥拥有者的特有编号（也可不选）。
- A_P：CA 的公钥和其他相关信息。
- T^A：数字认证的有效期限，包括此数字认证的启用日以及终止日。

X.509 对用户的认证另外规定了三种不同的程序。这些程序都用到了公开码技术。同时也假定双方都已从目录或都已交换了双方的证明文件。三种程序如下：

- （单向认证）单向认证的程序可以在一个回合的传输过程中，使接受者 Bob 识别由 Alice 产生的信息。t_A 是 Alice 设定的时间值，代表这份信息的有效期限，用来防止攻击者对此信息的延误。t_A 是防止"重放"（Replay）攻击的一个不重复的任意数。Bob 可以储存这个数值用来区分信息是否被"重放"。信息中也可以包含用 Alice 的密钥签署过的数据。对这份数据，Bob 可以用 Alice 的公开密钥鉴定其真伪。此外 Alice 更可以选定一只秘密通信密钥 K_{AB}，再用 Bob 的公开密钥将之加密 E_B（K_{AB}），从而与 Bob 共享此密钥，即



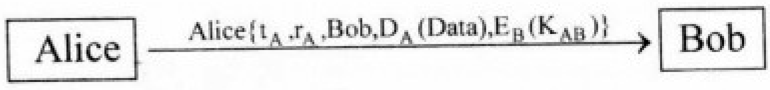
- （双向认证）"双向认证"，包含有两条信息，除了前述"单向认证"的发方送出的信息之外，接收方尚需产生一条回复信息给发送方。而其在安全性上达到的功能，除了前面所介绍的"单向识别"所能达到的三项认证功能外，尚可达到下列认证功能：

（1）由收方送回给发送方的身份识别资料，且可确认资料是接收方所产生的，且可以确认此信息的接收端确实是发送方。

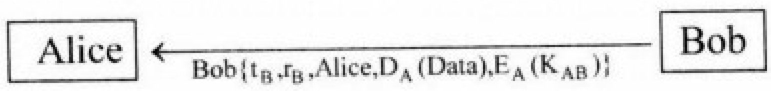
（2）可以保证此由接收方送回给发送方的身份识别资料的完整性。

（3）双方可以共享身份识别资料中秘密的部分（可选择是否采用）。

步骤 1：

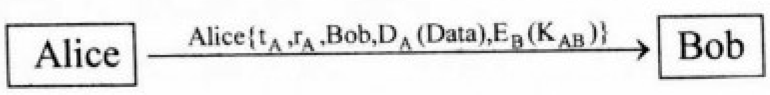


步骤 2：



- （三向认证）三向认证的识别程序较上述方法多了一道程序。多出的一道程序是 Alice 将 Bob 产生的随机数 r_B 解开后再传回给 Bob。这样的做法可以免除使用同步时钟值来区分信息是否被“重放”攻击。因为网络的时间延迟，时钟同步很难达成时，这样的识别方法非常有用。

步骤 1：



步骤 2：



步骤 3：



11.3　认证机构 CA

自然人认证（Citizen Digital Certificate），就是“电子身份证 IC 卡”，如图 11-1 所示。是公开密钥基础设施的一种应用（另外一个重要应用就是保健 IC 卡）。它是政府电子化的基础设施之一。据官方宣称，通过自然人认证，运用公开密钥密码技术，未来在政府与公民之间就可借助数字签名技术的应用，进行各种便捷的网络申办服务。大家只需要坐在电脑前，通过网络证明自己的身份（所以“自然人认证”也就是“网络上的身份证”），就可以免去以往舟车劳顿、大排长龙之苦，轻而易举地向各政府单位申请办理户籍本、网络报税等业务。自然人认证推广初期，将着重提供现有网络服务的业务，如户政、地政、税务、公路监理等，未来将有 1500 项政府业务开放上网申办，让大家足不出户就能办理各项业务，并提升政府行政效率与为民服务的品质，达到简政便民的政策目标。

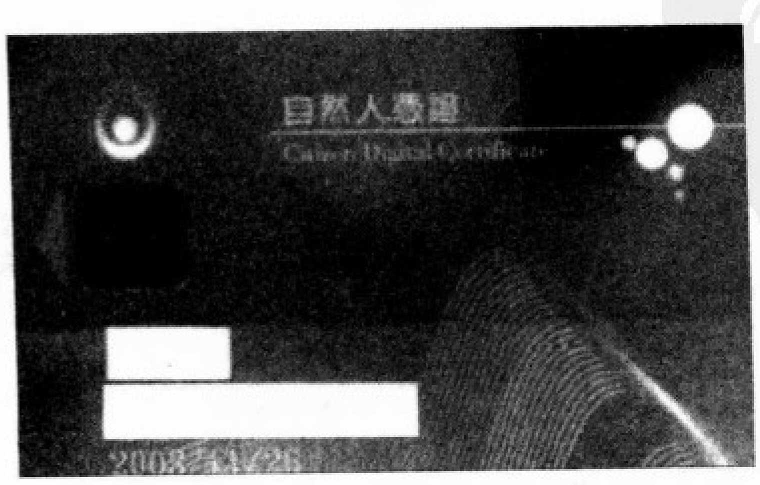


图 11-1　自然人认证

第 12 章　量子密码

量子密码学（Quantum Cryptography）大概是所有密码学相关领域中最让人难以理解的，但却也可能是密码加密算法中最后一个里程碑。这一切都要追溯到诺贝尔物理奖得主费曼（Richard Feylnnan）观察到的特定量子力学的现象，他发现这些现象是无法以传统电脑模拟的；然而他却反过头来建议，应该利用这些量子力学的特性，建构一种新的“量子电脑”，来大幅增加运算速度。当时并没有任何实例，也没有人知道如何去作，这在当时，只是大师的一个构想。一直到了 1994 年，量子计算才有了新的突破。在 AT&T 研究实验室工作的 Peter Shor 介绍了一种量子算法，这种量子算法可在多项式时间（概率式的）内分解整数，当然前提是量子电脑已经造好了；这意味着当代公开密钥密码 RSA 将会失效，而整个信息技术的理论平台，也必须将量子力学、量子计算纳入其中，注[1]甚至重新改写历史。

密码学的前卫技术，总是由国家安全单位秘密地发展采用，很可能对一般平民百姓还未搞清楚什么叫做数字认证的时候，国家安全机构量子电脑早已运转，而科学界的新发现，对他们而言，也只是后知后觉罢了。

12.1　量子实验

量子力学即使对物理学家而言也是困难的。量子力学的开创者之一波耳(Niels Bohr）也说道：“任何在思考量子力学时不会感到晕眩的人，都还没有了解它。”量子力学是处理粒子层面的物理学，光线的行为有时像是波，有时像是粒子，是具有波 / 粒子二重性质的，现代物理学认为一束光含有许多粒子，这些粒子，即光子（Photon），具有波一般的性质。任何人都可在家做以下的光子实验，如图 12-1 所示。

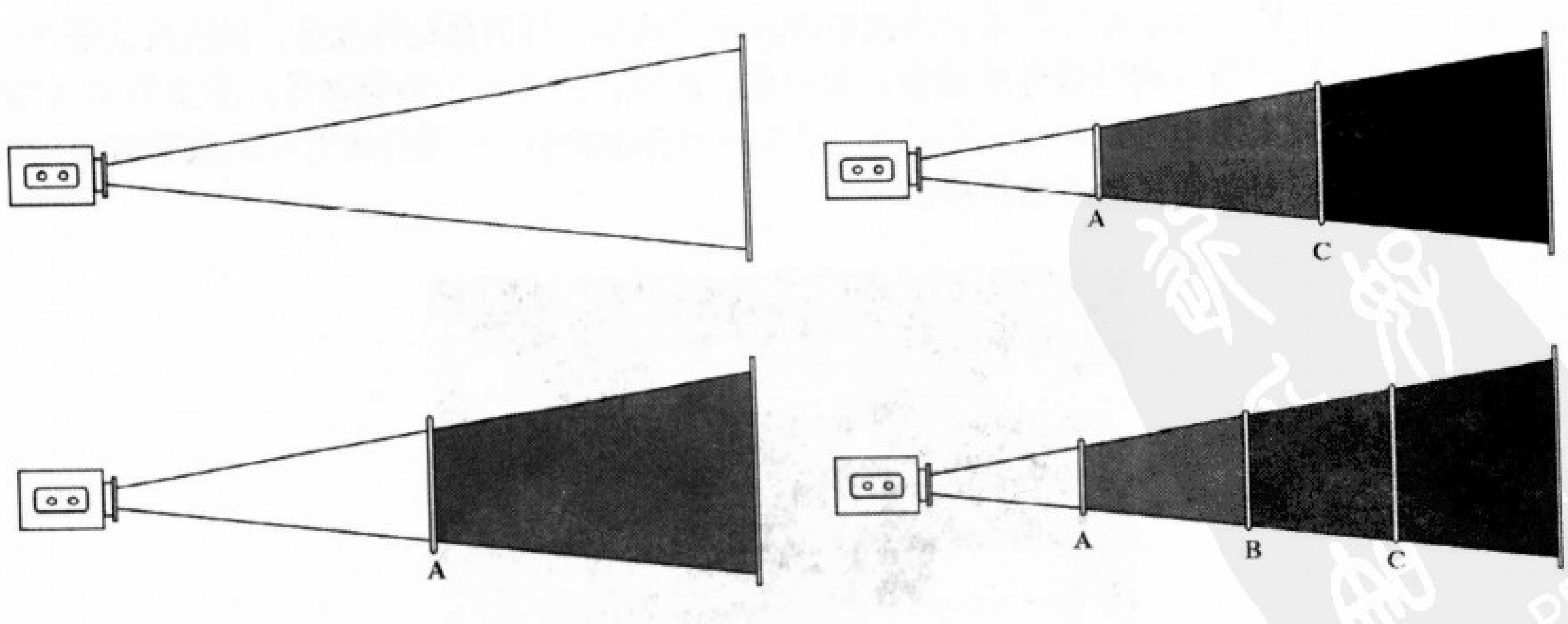


图 12-1　光子偏振实验

例 12.1（光子偏振）　考虑一束强光源照射在一投影幕布上，并准备三个偏光滤镜（Polaroid

Filter）以下同 A、B、C，偏光过滤方向分布为水平方向、45° 方向以及垂直方向，如图 12-1 所示。

（1）首先将偏光滤镜 A 放置于光束路径当中，发现幕布上亮度只剩下原亮度的一半。一种解释的方式就是每一个光子都有不同偏振方向，偏光滤镜 A 的作用就是过滤“特定”方向的光子。

（2）在光束路径上，介于偏光滤镜 A 与幕布之间，再放上偏光滤镜 C，此时幕布上已无任何光照，所有的光子，都被偏光滤镜 A 或偏光滤镜 C 挡住。

（3）此时再将偏光滤镜 B 放置于偏光滤镜 A 与偏光滤镜 C 当中，竟然发现，幕布上有些微的光芒，照度为最初照度的 1/8。

传统的经验论述或经典物理学无法解释这个现象，为何在偏光滤镜 A、C 当中再次放偏光滤镜 B 会有光照？

这可由光子偏振状态解释。每个光子都有偏振的状态，偏振的可能性是状态的叠加（Superposition of States）。

注 12.1 1933 年诺贝尔物理奖得主薛定谔（Erwin Schrödinger）曾以“薛定谔的猫”，用以解释叠加的概念：在盒子内的猫有两种状态：死的或活的。刚放入时猫还是活的，盒中放入一瓶剧毒氰化物，关上盖子的盒中的猫，不知其死活的，因为它可能舔氰化物或不舔，此时猫的状态就进入叠加状态，在打开盒子之际，观察猫，必处于其一种状态，非死即活，而状态叠加就告消逝。

这些可以用适当方向的单位向量表达，即任何偏振都可视为水平偏振向量 $|\rightarrow\rangle$ 与垂直偏振方向 $|\uparrow\rangle$ 的线性组合，即

$$a|\rightarrow\rangle + b|\uparrow\rangle。$$

由于偏振状态为单位向量，故系数 a，b 必须满足

$$|a|^2 + |b|^2 = 1，$$

而 a，b 也容许是复数（Complex Number），注[2]而偏光滤镜的作用就是执行“测量”光子的偏振：“测量”会导致两种输出结果，一种是与偏光滤镜平行方向，另一种是与偏光滤镜垂直方向。

（1）以水平方向偏光滤镜 A 测量光子，若光子偏振状态为 $a|\rightarrow\rangle + b|\uparrow\rangle$，有 $|a|^2$ 的概率会呈水平方向偏振通过，而 $|b|^2$ 的概率会呈垂直方向偏振被阻挡。因此对一束光子而言，会有 1/2 的光子呈水平方向偏振通过偏光滤镜 A。

（2）现在所有通过偏光滤镜 A 的光子都呈水平方向偏振，然而

$$|\rightarrow\rangle = \frac{1}{\sqrt{2}}|\nearrow\rangle + \frac{-1}{\sqrt{2}}|\nwarrow\rangle，$$

因此，经过 45° 方向偏光滤镜测量，剩下 $\left(\frac{-1}{\sqrt{2}}\right)^2 = \frac{1}{2}$ 的光子，呈↖方向偏振被阻挡，剩下的 $\frac{1}{2}$ 的光子会呈↗方向偏振通过，此时剩下的 $\frac{1}{4}$ 光子呈↗方向偏振通过偏光滤镜 A 及 B。

（3）所有通过偏光滤镜 B 的光子都呈↗方向偏振，而

$$|\nearrow\rangle=\frac{1}{\sqrt{2}}|\uparrow\rangle+\frac{1}{\sqrt{2}}|\rightarrow\rangle,$$

故通过垂直方向偏光滤镜 C 的光子都呈垂直方向偏振，而只剩下最初光子总数的$\frac{1}{2}\times\frac{1}{4}=\frac{1}{8}$，其余的都分别被三个偏光滤镜挡下。

12.2 量子密钥分配

上节所说的光子偏振状态，其实就是量子计算的基础，要建构一量子电脑，就要先了解什么是量子电脑的最基本单量元（Quantum bit 或 Qubit）：

定义 12.1（量元，Qubit） 一个 Qubit 即在二维复数向量空间的单位向量，其中固定一正交基底以

$$\{|0\rangle,|1\rangle\}$$

表示（可以是$\{|\rightarrow\rangle,|\uparrow\rangle\}$或$\{|\nearrow\rangle,|\nwarrow\rangle\}$），与古典位概念不同，一 Qubit 其实是以“叠加”（Superposition）的状态呈现，如

$$a|0\rangle+b|1\rangle,$$

其中 a，b 为复数且$|a|^2+|b|^2=1$。

如同光子偏振情形，若此叠加状态以正交基底$\{|0\rangle,|1\rangle\}$测量，其测量值为$|0\rangle$的概率为$|a|^2$，而测量值为$|1\rangle$的概率为$|b|^2$。

Alice 与 Bob 如何利用此来加密传递信息？他们需要建立两个通道（Channel）：一个是传统通道、另一个是量子通道。（通过量子通道的光子不会改变其光子偏振状态的，传统通道是双向的，而量子通道是单向的）。其密钥分配的做法整理如下。

算法 12.1（量子密钥分配） Alice 想与 Bob 使用量子密钥分配约定出一传统密码的密钥。

- Alice 用传统通道传一组 $\mathbb{F}_2$ 字串

  $$b_1b_2b_3b_4\cdots b_n$$

  给 Bob。
- Alice 选择两组正交基底

  $$\oplus=\{|\uparrow\rangle,|\rightarrow\rangle\}\text{ 以及 }\oplus=\{|\nwarrow\rangle,|\nearrow\rangle\}。$$
- Alice 将字串 $b_1b_2\cdots b_n$ 随机编码呈光子偏振状态：

  $$\begin{cases}0\rightarrow|\uparrow\rangle, & \text{若用}\oplus\\ 1\rightarrow|\rightarrow\rangle, & \text{若用}\oplus\\ 0\rightarrow|\nwarrow\rangle, & \text{若用}\otimes\\ 1\rightarrow|\nearrow\rangle, & \text{若用}\otimes\end{cases}$$

  将这些光子偏振状态利用量子通道传给 Bob。
- Bob 测量随机以正交基底$\oplus$或$\otimes$测量这些光子偏振，并秘密地将这些测量结果记录下来。
- Bob 利用传统通道，告知 Alice，每次测量光子所用的基底。

- Alice 比对她与 Bob 对每次测量光子所用的基底选择，并利用传统通道告知 Bob 哪些光子的测量是用相同的基底。而这些相同位便是所协定的传统密码所用的密钥。

例 10.2　假设 Alice 与 Bob 欲利用量子密钥分配约定单次密码本的密钥，其过程如下：

（1）Alice 利用传统通道，传位

0111001110，

给 Bob。

（2）Alice 用基底

⊕，⊗，⊗，⊕，⊗，⊗，⊕，⊗，⊗，⊕，如表 12-1 所示。

表 12.1　例 12.2 中 Alice 与 Bob 使用量子密钥分配位资料与量元资料比对

Alice 用传统通路传递信息	0	1	1	1	0	0	1	1	1	0
Alice 选用基底	⊕	⊗	⊗	⊕	⊗	⊗	⊕	⊗	⊗	⊕
Alice 用量子通路传递信息	|↑⟩	|↗⟩	|↗⟩	|→⟩	|↖⟩	|↖⟩	|→⟩	|↗⟩	|↗⟩	|↑⟩
Bob 选用基底	⊕	⊕	⊗	⊗	⊕	⊗	⊕	⊗	⊕	⊗
产生密钥	0		1			0	1	1		

（3）Alice 利用量子通道传递 Qubit

|↑⟩，|↗⟩，|↗⟩，|→⟩，|↖⟩，|↖⟩，|→⟩，|↗⟩，|↗⟩，|↑⟩

给 Bob。

（4）Bob 选择基底

⊕，⊕，⊗，⊗，⊕，⊗，⊕，⊗，⊕，⊗

测量 Qubit，并利用传统通道告知 Alice 他使用哪个基底。

（5）Alice 比对发现相符的基底选择，即第 1、3、6、7、8 选择相符，并利用传统通道告知 Bob，所以约定之密钥值为 01011。

量子密钥分配的安全性是建立在量子力学的基本定律：

"只要测量光子，就会引起光子偏振状态的改变。"

攻击者 Eve 要截收 Alice 与 Bob 之间的通信信息，就必须测量在量子通道上所传送的光子。假设 Eve 测量 Alice 所传送的光子偏振，因经过测量，光子偏振状态可能改变（只要 Eve 用错基底，这出错的概率有 50%）。当 Bob 测量这些已经 Eve 测量的光子，Bob 有 25%的机会测量错误。举例而言，若 Alice 传 Qubit|→⟩ 给 Bob，Alice 与 Bob 均使用基底⊕，假设 Eve 在测量此 Qubit 时用基底⊗，所得的测量值为|↗⟩或|↖⟩（概率各半），将此测量过的光子传给 Bob，Bob 测量光子偏振值为|→⟩或|↑⟩（概率也各半），因此在此出错的概率就有 25%，如此的错误，可借助 Alice 所传的位值检查出来，而 Eve 测量二个光子，会产生错误的概率为 $1-\left(\frac{3}{4}\right)^n$，毫无疑问，只要所测量光子数 n 足够多，出错的概率就越接近 1。如此利用量子密钥分配与单次加密本法，就可建立一个无条件安全的密码系统，此时就算 Eve 再有能耐，也无法破译任何密文，或许这是最最理想的密码系统。值得一提的是，在 1995 年，日内瓦大学的研究员在相距 24km 的日内瓦和 Nyon 通过一光纤管，成功地执行了量子密码。注[3]以量子电脑与量子密码比较，量子密码的技术是较为成熟的，除了瑞士的实例，甚至美国目前所进行的量子密码计

划，打算建立一套能以人造卫星运作的量子密码系统，目前已能在大气层中跨越 1km 的距离传输一只量子密钥。或许如许多专家所预测的，不需要太久，量子密码就可以成为实用的技术，若真如此，或许密码技术的发展将画上一体止符；另外如果量子电脑技术成熟，那么真正实用安全的密码系统大概也只剩下量子密码了。

12.3 浅谈 Shor 之量子算法

最成功的量子算法应属 Shor 所提出的算法，注[4]诚然这是一个概率式的算法，可以用来分解 RSA 模数 n，也可以略加修改攻击所有以离散对数为基础的公开密钥密码系统，以及数字签名系统，除非运气很差，一般都可在多项式时间内破译成功；然而量子算法要能发挥威力，是要在量子电脑造好以后。倘若量子电脑能真正研发成功，所有截至目前所发展的公开密钥密码主流系统，如 RSA、ElGamal，或是椭圆曲线密码系统，都将成为历史，所有建立在公开密钥密码技术上的实际应用，都必须重新检视其适用性，整个密码学发展、相关的信息安全领域，都将被颠覆，因为一旦到了量子电脑技术成熟之时，大概只剩下量子密码是安全的。这并非危言耸听，截至目前为止，我们仅能从科学界、业界略知这方面的发展，至于国家安全单位的成果，大概也只有他们自己知道，真正的发展如何，还是一团迷雾。

以质因数分解大整数 n 而言，传统最快的分解法是代数数域筛法，其计算复杂度为

$$\exp(O((\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}))\text{。}$$

Shor 的基本想法是要将 n 的因数分解问题化成模指数函数的周期问题；随机取一个正整数 a，其中$1<a<n$，可利用广义辗转相除法或 xeuclidean()计算 a 与 n 的最大公因数，即 $\gcd(a,n)$，若$\gcd(a,n)>1$，那就能将 n 因数分解，不过绝大部分的 a 都会与 n 互质。根据 Euler 定理，存在某正整数 r 使得

$$a^r \equiv 1 \pmod n\text{，}$$

其中 a 与 n 互质，而如此最小的 r 是 a 在乘法群中 $\mathbb{Z}/n^{\times}$ 的秩（即模指数函数之周期），而 Shor 算法可以计算 a 的秩 r 值。

在这种情况下，若 r 为偶数，其实可将上式改写成

$$(a^{r/2}-1)(a^{r/2}+1)\equiv 0 \pmod n\text{；}$$

令$\alpha=a^{r/2}-1$，$\beta=a^{r/2}+1$，n 必然整除$\alpha\beta$；若α以及β都不是 n 的倍数，如此计算

$$\gcd(\alpha,n)\text{ 与 }\gcd(\beta,n)$$

就可得 n 的因子。

例 12.3 令$n=15$，取一整数$a=7$，计算 7 的幂次方得

$$7\equiv 7 \pmod{15}$$

$$7^2\equiv 4 \pmod{15}$$

$$7^3\equiv -2 \pmod{15}$$

$$7^4\equiv 1 \pmod{15}\text{，}$$

因此 7 在 $\pmod{15}$ 运算的秩为 4，即

$$7^4 \equiv 1 \pmod{15}$$
$$\Rightarrow (7^{4/2}-1)(7^{4/2}+1) = 48 \times 50。$$

计算

$$\gcd(48,15)=3、\gcd(50,15)=5$$

得 $n=15$ 的因数 3,5。

如此找因数的方法是寄希望于 $(a^{r/2}\pm 1)$ 两整数都不是 n 的倍数，然而这并不是都成立的，但是满足如此条件的 a 的概率却是 $\geqslant \frac{1}{2}$。

定理 12.1　令 n 为奇数，且 $a<n$ 是一与 n 互质的随机整数。令 r 为 a 在乘法群 $\mathbb{Z}/n^{\times}$ 的秩。则 r 为偶数且同时 $a^{r/2}\pm 1$ 皆非 n 的倍数的概率为 $\geqslant \frac{1}{2}$。

因此，只要重复以上步骤 k 次，能将 n 因数分解的概率 $\geqslant 1-\frac{1}{2^k}$，如此又是一个 Las Vegas 算法，这部分计算主要用到广义辗转相除法，在多项式时间内可完成。

而函数

$$f(x) = a^x \pmod{n}$$

的周期计算，在 Shor 算法中要用到量子傅立叶变换（Quantum Fourier Transform）以及连分数逼近，后者已在本书 3.8 节“连分数”中介绍过，而前者量子傅立叶子变换，其实是快速傅立叶变换（Fast Fourier Transform）的量子计算版，需用到线性代数中张量积（Tensor Product）等概念，而这部分的计算，也可在多项式时间内完成（不过必须是在量子电脑上执行），因此在量子电脑上，只须多项式时间的因式分解算法，就可完成。

参考文献

[1] Agrawal, M.; Kayal, M. and Saxena, N., "*PRIMES is in P*", Preprint of Aug,2002

[2] Bishop, David, "*Introduction to Cryptography with Java Applets*", Jones and Bartlett, 2003

[3] Blake, J. F.; Seroussi, G. and Smart, N. P., "*Elliptic Curves in Cryptography.*" Cambridge University Press, p. 161, 1999

[4] Blum, L.; Blum, M. and Shub, M., "*A Simple Unpredictable Pseudo-Random Number Generator*", SIAM Journal on Computing, No. 2, 1986

[5] Boneh, Dan, "*Twenty years of Attacks on the RSA Cryptosystem.*" Notices of AMS, 46, p. 203-213, 1999

[6] Bornemann, F., "*PRIMES Is in P: A Breakthrough for 'Everyman'* " Notices of the AMS, May 2003

[7] Buchmann, Johannes A., "*Introduction to Cryptography*", Springer, 1999

[8] Campbell, K. W. and Wiener, M. T., "*DES IS Not a Group*" , Advances in Cryptology-CRYPTO'92 Proceedings, Springer-Verlag, p. 512-520

[9] Cohen, Daniel I. A., "*Introduction to Computer Theory*", 2nd ed., John Wiley & Sons, Inc., 1997

[10] Cohen, Henri, "*A Course in Computational Algebraic Number Theory,*" Springer-Verlag, Berlin, Heidelberg, 1993

[11] Cohen, H.; Miyaji, A. and Ono, T., "*Efficient elliptic curve exponentiation using mixed coordinates*". ASIACRYPT 98. Springer-Verlag, LNCS 917, 51-65, 1998

[12] Churchhouse, Robert, "*codes and ciphers*", Cambridge, 2002

[13] Daemen, J. and Rijmen, V., "*Rijndael: The Advanced Encryption Standard*", Dr. Dobb's Journal, March 2001

[14] Dahms, H. G¨unther, "*Der Zweite Weltkrieg in Text und Bild*", Herbig, 1995

[15] Diffie and Hellman, "*New directions in cryptography.*" *IEEE Trans. Inform. Theory IT-22*, p. 644-654, 1976

[16] Eker, A. and Jozsa, R., "*Quantum computation and Shor's factoring algorithm,*" *Reviews of Modern Physics.*, 68, 733, p. 733-753, 1996

[17] ElGamal, T., "*A public key cryptosystem and signature scheme based on discrete logarithms*", IEEE Trans. Inf. theory. 31, p. 469-472, 1985

[18] Elkies, N. D., "*Elliptic and modular curves over finite field and related computational issues*", Computational Perspectives on Number Theory: Proc. Conf.in Honor of A.O.L. Atkin, AMS. Internat. Press, 7, 21-76, 1998

[19] Ellis, J., "*The Possibility of Secure Non-Secret Digital Encryption*", CESG Report, January 1970

[20] Ellis, J., "*The History of Non-Secret Encryption*", Cryptologia, July 1999

[21] Garrett, Paul, "*Making, Breaking Codes: an introduction to cryptography*", Prentice-Hall, 2001

[22] Gordan, John A., "*Strong primes are easy to find*", in *Proceedings of EUROCRYPT 84*,, p. 216-223, Paris, Springer-Verlag,LNCS. 209., 1985

[23] Hughes, R. J.; Buttler, W. T.; Kwiat, P. G.; Luther, G. G.; Morgan, G. L.; Nordholt, J. E.; Peterson, C. G. and Simmon, C. M., " Secure communications using quantum cryptography" in *Photonic Quantum Computing*, Vol3076, p.2-11, 1997

[24] IEEE P1363/D3(Draft Version 3), Shandard specifications for public key cryptography,May 1998

[25] Kahn, D., "*The Codebreakers: The Story of Secret Writing*", New York, cribner, 1996

[26] Kippenhahn, Rudolf, "*Verschl¨usselte Botschaften*", G¨ottingen, 1997

[27] Kocher, P., "*Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*", CRYPTO '96, Springer-Verlag, LNCS 1109, 104-113,1996

[28] Knuth, "*The Art of Computer Programming*", Vol 2, 3rd ed., Addison-Wesley, 1998

[29] Koblitz, N., "*Elliptic Curve cryptosystems.*" Mathematics of Computation 48, p. 203-209, 1987

[30] Koblitz, N., "*CM-curves with good cryptographic properties*", CRYPTO 91, Springer-Verlag, LNCS 576, 279-287, 1992

[31] Koblitz, N., "*Algebraic aspects of cryptography. 3, Algorithms and Computation in Mathematics*", Springer-Verlag, 1998

[32] Kocher, P., "*Timing Attacks on Implementations of Diffie-Helman, RSA, DSS, and Other Systems*", Proceedings, Crypto '96, August 1996

[33] Lang, S., "*Elliptic Curves: Diophantine Analysis.*" Springer-Verlag, 1978

[34] Lenstra, H. W., "*Factoring integers with elliptic curves.*" Annals of Math., 126, p. 649-673, 1987

[35] Lidl, R. and Niederreiter, H., "*Finite Fields*", Encyclopedia of Mathematics and its Application, G.C. Rota, editor, Addison-Wesley, 1983

[36] Menezes, A. J., "*Elliptic Curve Public Key Cryptosystems*". Kluwer Academic Publishers, 1993

[37] Menezes, A. J.; van Oorschot, P. C. and Vanstone, S.A., "*Handbook of Applied Cryptography.*" CRC Press, 1996

[38] Menezes, A. J.; Okamoto, T. and Vanstone, S. A., "*Reducing elliptic curve logarithms to a finite field.*", IEEE Trans. Info. Theory, 39, 1639-1646, 1993

[39] Menezes, A. J.; Vanstone, S. A. and Zuccherato, R. J., "*Counting points on elliptic curves over Fn*2 ", Math. Comp., 60, 407-420, 1993

[40] Meyer, B. and M¨uller, V., "*A public key cryptosystem based on elliptic curves over Z=nZ equivalent to factoring*", EUROCRYPT 91. Springer-Verlag, LNCS 1070, 49-59, 1996

[41] Miller, G., "*Riemann's hypothesis and test for primality*", J. Comp. and Sys. Sci, 13, 300-317,

1976

[42] Miller, "*Use of elliptic curve in cryptography.*" Advances in Cryptology, CRYPTO 85. " Springer-Verlag, LNCS 218, p.417-426, 1986

[43] Mollin, Richard A., "*RSA and Public-Key Cryptography*", Chapman & Hall/CRC, 2002

[44] Montgomery, P. L., "*Speeding the Pollard and elliptic curve methods of factorization.*",Math. Comp., 48, 243-264, 1987

[45] Montgomery, Peter L.,"*Modular mulliplication without trial division*", Mathematics of Computation, p. 519-521, 44(170), 1985

[46] van Oorschot, P. and Wiener, M., "*Parallel Collision Search with Application to Hash Functions and Discrete Logarithms.* ", Proc. 2nd ACM Conf. On Computer a. Communicat. Sec., 1994

[47] Pollard, J. M., "*Theorems on factorization and primality testing.*" Proccedings Cambridge Philosphical Society,76, p. 521-528, 1974

[48] Pohlig, S.C. and Hellman, M.E., "*An improved algorithm for computing discrete logarithms over GF(p) and its cryptographic significance*", IEEE-IT, vol.24 (1978), p. 106-110

[49] Rabin, M., "*Digitized signatures and public key functions as intractable as factorization*",MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979

[50] Rejewski, Marian, "*An Application of the Theory of Permutations in Breaking the Enigma Cipher*", Applicationes Mathematicae. 16, No. 4, Warsaw, 1980

[51] Rivest, Shamir and Adleman, "*A Method for Obtaining Digital Signature and Public-Key Cryptosystems*", Communication of the A.C.M., 21. No.2 p. 120-126, 1978

[52] Rivest, R. and Silverman, Robert D., "*Are 'Strong' Primes Needed for RSA?*" IACR Cryptology, 2001

[53] Rosen, Kenneth H., "*Elementary Number Theory and its application*", Addison Wesley Longman, Inc., 2000

[54] Rosing, Michael, "*Implemting Elliptic Curve Cryptography*", Greenwich, Manning, 1999

[55] Schneier, Bruce, "*Applied Cryptography*", John Wiley & Sons, Inc., 1996

[56] Schoof, R., "*Counting points on elliptic curves over finite fields*", in J. Th´eorie des Nombres de Bordeaux, 7, 219-254, 1995

[57] Shor, Peter, "*Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.*" Society for Industrial and Applied Mathematics Journal on Computing , 26, 5, 1484-1509, 1997 (http://xxx.lanl.gov/abs/quant-ph/9508027)

[58] Schroeder, M. R., "*Number Theory in Science and Communication*", Springer-Verlag, 1983

[59] Shannon, Claude, "*A mathematical theory of communication*", Bell Systems Technical Journal, 27, p. 379-423, 1948

[60] Shannon, Claude, "*Communication theory of secrecy systems*", Bell Systems Technical Journal,28, p. 656-715, 1949

[61] Silverman, Josef H., "*The Arithmetic of Elliptic Curves*", Springer-Verlag,1985

[62] Silverman, Josef H., "*Advanced Topics in the Arithmetic of Elliptic Curves.*", in

Springer-Verlag, GTM 151, 1994

[63] Singh, Simon, "*The Code Book*", 2000

[64] Smart, Nigel, "*Cryptography: An Introduction*", McGraw-Hill, 2003

[65] Stallings, William, "*Cryptography and Netwrok Security: Principles and Practice, 3rd ed.*", Pearson Education International, 2003

[66] Stinson, Douglas R., "*Cryptography: Theory and Practice*", Chapman & Hall/CRC, 2002

[67] Trappe, Wade and Washington, Lawrence C. , "*Introduction to Cryptography with Coding Theory*", Prentince Hall, 2002

[68] van Oorschot, Paul C. and Wiener, Michael, J., "*Parallel collision search with cryptanalytic applications*", Journal of Cryptography, 12, p. 1-28, 1999

[69] Welschenbach, Michael, "*Cryptography in C and C++*", Apress, 2001

[70] Wiener, M., "*Cryptanalysis of short RSA secret exponents.*", IEEE Trans. Inform. Theory, 36, p. 553-558, 1990

[71] Wiles, A., "*Modular elliptic curves and Fermat's Last Theorem.*", Ann Math., 142, 443-551, 1995

[72] Williams, H. C., "*A p+1 method for factoring.*" Mathematics of Computation, 39(159), p. 225-234, 1982

[73] Zimmerman, P., "*The Official PGP User's Guide*", MIT Press, Cambridge,1995

[74] http://www.bletchleypark.org.uk

[75] http://csrc.nist.gov/crypto/toolkit/rijndael

[76] http://www.rsasecurity.com/rsalabs/challenges/des3/index.html

[77] http://www.cse.iitk.ac.in/news/primality.html

[78] http://java.sun.com/j2se/1.3/docs/api/java/math/BigInteger.html

[79] http://www.informatik.tu-darmstadt.de/TI/LiDIA

[80] http://www.rsasecurity.com/rsalabs/pkcs/

[81] http://www.turing.org.uk/turing

[82] http://www.cesg.gov.uk/publications/index.htm#nsecret